

# **.NET Test Automation Recipes**

A Problem-Solution Approach



James D. McCaffrey

## **.NET Test Automation Recipes: A Problem-Solution Approach**

**Copyright © 2006 by James D. McCaffrey**

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

ISBN-13: 978-1-59059-663-0

ISBN-10: 1-59059-663-3

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Trademarked names may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, we use the names only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

Lead Editor: Jonathan Hassell

Technical Reviewer: Josh Kelling

Editorial Board: Steve Anglin, Ewan Buckingham, Gary Cornell, Jason Gilmore, Jonathan Gennick,

Jonathan Hassell, James Huddleston, Chris Mills, Matthew Moodie, Dominic Shakeshaft, Jim Sumser,

Keir Thomas, Matt Wade

Project Manager: Elizabeth Seymour

Copy Edit Manager: Nicole LeClerc

Copy Editor: Julie McNamee

Assistant Production Director: Kari Brooks-Copony

Production Editor: Katie Stence

Composer: Lynn L'Heureux

Proofreader: Elizabeth Berry

Indexer: Becky Hornak

Cover Designer: Kurt Krames

Manufacturing Director: Tom Debolski

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail [orders-ny@springer-sbm.com](mailto:orders-ny@springer-sbm.com), or visit <http://www.springeronline.com>.

For information on translations, please contact Apress directly at 2560 Ninth Street, Suite 219, Berkeley, CA 94710. Phone 510-549-5930, fax 510-549-5939, e-mail [info@apress.com](mailto:info@apress.com), or visit <http://www.apress.com>.

The information in this book is distributed on an "as is" basis, without warranty. Although every precaution has been taken in the preparation of this work, neither the author(s) nor Apress shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this work.

The source code for this book is available to readers at <http://www.apress.com> in the Source Code section.

# Contents at a Glance

About the Author .....	xiii
About the Technical Reviewer .....	xv
Acknowledgments .....	xvii
Introduction .....	xix
<b>CHAPTER 1</b> API Testing .....	3
<b>CHAPTER 2</b> Reflection-Based UI Testing .....	33
<b>CHAPTER 3</b> Windows-Based UI Testing .....	65
<b>CHAPTER 4</b> Test Harness Design Patterns .....	97
<b>CHAPTER 5</b> Request-Response Testing .....	135
<b>CHAPTER 6</b> Script-Based Web UI Testing .....	167
<b>CHAPTER 7</b> Low-Level Web UI Testing .....	185
<b>CHAPTER 8</b> Web Services Testing .....	207
<b>CHAPTER 9</b> SQL Stored Procedure Testing .....	237
<b>CHAPTER 10</b> Combinations and Permutations .....	265
<b>CHAPTER 11</b> ADO.NET Testing .....	301
<b>CHAPTER 12</b> XML Testing .....	335
<b>INDEX</b> .....	365

# Contents

About the Author .....	xiii
About the Technical Reviewer .....	xv
Acknowledgments .....	xvii
Introduction .....	xix

## PART 1 ■ ■ ■ Windows Application Testing

■ CHAPTER 1	<b>API Testing</b> .....	3
	1.0 Introduction .....	3
	1.1 Storing Test Case Data .....	6
	1.2 Reading Test Case Data .....	7
	1.3 Parsing a Test Case .....	8
	1.4 Converting Data to an Appropriate Data Type .....	9
	1.5 Determining a Test Case Result .....	11
	1.6 Logging Test Case Results .....	13
	1.7 Time-Stamping Test Case Results .....	16
	1.8 Calculating Summary Results .....	17
	1.9 Determining a Test Run Total Elapsed Time .....	19
	1.10 Dealing with null Input/null Expected Results .....	20
	1.11 Dealing with Methods that Throw Exceptions .....	22
	1.12 Dealing with Empty String Input Arguments .....	24
	1.13 Programmatically Sending E-mail Alerts on Test Case Failures ...	26
	1.14 Launching a Test Harness Automatically .....	28
	1.15 Example Program: ApiTest .....	29

<b>CHAPTER 2</b>	<b>Reflection-Based UI Testing</b>	33
	2.0 Introduction	33
	2.1 Launching an Application Under Test	35
	2.2 Manipulating Form Properties	39
	2.3 Accessing Form Properties	44
	2.4 Manipulating Control Properties	47
	2.5 Accessing Control Properties	50
	2.6 Invoking Methods	53
	2.7 Example Program: ReflectionUITest	58
<b>CHAPTER 3</b>	<b>Windows-Based UI Testing</b>	65
	3.0 Introduction	65
	3.1 Launching the AUT	66
	3.2 Obtaining a Handle to the Main Window of the AUT	68
	3.3 Obtaining a Handle to a Named Control	73
	3.4 Obtaining a Handle to a Non-Named Control	75
	3.5 Sending Characters to a Control	78
	3.6 Clicking on a Control	80
	3.7 Dealing with Message Boxes	82
	3.8 Dealing with Menus	86
	3.9 Checking Application State	89
	3.10 Example Program: WindowsUITest	91
<b>CHAPTER 4</b>	<b>Test Harness Design Patterns</b>	97
	4.0 Introduction	97
	4.1 Creating a Text File Data, Streaming Model Test Harness	100
	4.2 Creating a Text File Data, Buffered Model Test Harness	104
	4.3 Creating an XML File Data, Streaming Model Test Harness	108
	4.4 Creating an XML File Data, Buffered Model Test Harness	113
	4.5 Creating a SQL Database for Lightweight Test Automation Storage	117
	4.6 Creating a SQL Data, Streaming Model Test Harness	119
	4.7 Creating a SQL Data, Buffered Model Test Harness	123
	4.8 Discovering Information About the SUT	126
	4.9 Example Program: PokerLibTest	129

## PART 2 ■ ■ ■ Web Application Testing

■ CHAPTER 5	<b>Request-Response Testing</b> .....	135
	5.0 Introduction .....	135
	5.1 Sending a Simple HTTP GET Request and Retrieving the Response .....	138
	5.2 Sending an HTTP Request with Authentication and Retrieving the Response .....	139
	5.3 Sending a Complex HTTP GET Request and Retrieving the Response .....	140
	5.4 Retrieving an HTTP Response Line-by-Line .....	141
	5.5 Sending a Simple HTTP POST Request to a Classic ASP Web Page .....	143
	5.6 Sending an HTTP POST Request to an ASP.NET Web Application ...	145
	5.7 Dealing with Special Input Characters .....	150
	5.8 Programmatically Determining a ViewState Value and an EventValidation Value .....	152
	5.9 Dealing with CheckBox and RadioButtonList Controls .....	156
	5.10 Dealing with DropDownList Controls .....	157
	5.11 Determining a Request-Response Test Result .....	159
	5.12 Example Program: RequestResponseTest .....	162
■ CHAPTER 6	<b>Script-Based Web UI Testing</b> .....	167
	6.0 Introduction .....	167
	6.1 Creating a Script-Based UI Test Harness Structure .....	170
	6.2 Determining Web Application State .....	172
	6.3 Logging Comments to the Test Harness UI .....	173
	6.4 Verifying the Value of an HTML Element on the Web AUT .....	174
	6.5 Manipulating the Value of an HTML Element on the Web AUT .....	176
	6.6 Saving Test Scenario Results to a Text File on the Client .....	177
	6.7 Saving Test Scenario Results to a Database Table on the Server ..	179
	6.8 Example Program: ScriptBasedUITest .....	181

<b>CHAPTER 7</b>	<b>Low-Level Web UI Testing</b> .....	185
	7.0 Introduction .....	185
	7.1 Launching and Attaching to IE .....	188
	7.2 Determining When the Web AUT Is Fully Loaded into the Browser .	190
	7.3 Manipulating and Examining the IE Shell .....	192
	7.4 Manipulating the Value of an HTML Element on the Web AUT .....	194
	7.5 Verifying the Value of an HTML Element on the Web AUT .....	195
	7.6 Creating an Excel Workbook to Save Test Scenario Results .....	198
	7.7 Saving Test Scenario Results to an Excel Workbook .....	200
	7.8 Reading Test Results Stored in an Excel Workbook .....	201
	7.9 Example Program: LowLevelUITest .....	203
<b>CHAPTER 8</b>	<b>Web Services Testing</b> .....	207
	8.0 Introduction .....	207
	8.1 Testing a Web Method Using the Proxy Mechanism .....	212
	8.2 Testing a Web Method Using Sockets .....	214
	8.3 Testing a Web Method Using HTTP .....	220
	8.4 Testing a Web Method Using TCP .....	222
	8.5 Using an In-Memory Test Case Data Store .....	226
	8.6 Working with an In-Memory Test Results Data Store .....	229
	8.7 Example Program: WebServiceTest .....	232

## PART 3 ■ ■ ■ Data Testing

<b>CHAPTER 9</b>	<b>SQL Stored Procedure Testing</b> .....	237
	9.0 Introduction .....	237
	9.1 Creating Test Case and Test Result Storage .....	239
	9.2 Executing a T-SQL Script .....	241
	9.3 Importing Test Case Data Using the BCP Utility Program .....	243
	9.4 Creating a T-SQL Test Harness .....	245
	9.5 Writing Test Results Directly to a Text File from a T-SQL Test Harness .....	249
	9.6 Determining a Pass/Fail Result When the Stored Procedure Under Test Returns a Rowset .....	252
	9.7 Determining a Pass/Fail Result When the Stored Procedure Under Test Returns an out Parameter .....	254
	9.8 Determining a Pass/Fail Result When the Stored Procedure Under Test Does Not Return a Value .....	256
	9.9 Example Program: SQLspTest .....	259

<b>CHAPTER 10</b>	<b>Combinations and Permutations</b>	265
10.0	Introduction	265
10.1	Creating a Mathematical Combination Object	267
10.2	Calculating the Number of Ways to Select k Items from n Items	269
10.3	Calculating the Successor to a Mathematical Combination Element	271
10.4	Generating All Mathematical Combination Elements for a Given n and k	273
10.5	Determining the mth Lexicographical Element of a Mathematical Combination	275
10.6	Applying a Mathematical Combination to a String Array	278
10.7	Creating a Mathematical Permutation Object	280
10.8	Calculating the Number of Permutations of Order n	282
10.9	Calculating the Successor to a Mathematical Permutation Element	284
10.10	Generating All Mathematical Permutation Elements for a Given n	286
10.11	Determining the kth Lexicographical Element of a Mathematical Permutation	287
10.12	Applying a Mathematical Permutation to a String Array	291
10.13	Example Program: ComboPerm	293
<b>CHAPTER 11</b>	<b>ADO.NET Testing</b>	301
11.0	Introduction	301
11.1	Determining a Pass/Fail Result When the Expected Value Is a DataSet	303
11.2	Testing a Stored Procedure That Returns a Value	306
11.3	Testing a Stored Procedure That Returns a Rowset	309
11.4	Testing a Stored Procedure That Returns a Value into an out Parameter	311
11.5	Testing a Stored Procedure That Does Not Return a Value	314
11.6	Testing Systems That Access Data Without Using a Stored Procedure	318
11.7	Comparing Two DataSet Objects for Equality	321
11.8	Reading Test Case Data from a Text File into a SQL Table	324
11.9	Reading Test Case Data from a SQL Table into a Text File	327
11.10	Example Program: ADOdotNETtest	329



■ <b>CHAPTER 12 XML Testing</b> .....	335
12.0 Introduction .....	335
12.1 Parsing XML Using XmlTextReader .....	337
12.2 Parsing XML Using XmlDocument .....	339
12.3 Parsing XML with XPathDocument .....	341
12.4 Parsing XML with XmlSerializer .....	343
12.5 Parsing XML with a DataSet Object .....	347
12.6 Validating XML with XSD Schema .....	350
12.7 Modifying XML with XSLT .....	353
12.8 Writing XML Using XmlTextWriter .....	355
12.9 Comparing Two XML Files for Exact Equality .....	356
12.10 Comparing Two XML Files for Exact Equality, Except for Encoding .....	358
12.11 Comparing Two XML Files for Canonical Equivalence .....	359
12.12 Example Program: XmlTest .....	361
■ <b>INDEX</b> .....	365

# About the Author

■ **DR. JAMES MCCAFFREY** works for Volt Information Sciences, Inc. He holds a doctorate from the University of Southern California, a master's in information systems from Hawaii Pacific University, a bachelor's in mathematics from California State University at Fullerton, and a bachelor's in psychology from the University of California at Irvine. He was a professor at Hawaii Pacific University, and worked as a lead software engineer at Microsoft on key products such as Internet Explorer and MSN Search.

# About the Technical Reviewer

■ **JOSH KELLING** is a private consultant working in the business software industry. He is formally educated in physics and self-taught as a software developer with nearly 10 years of experience developing business and commercial software using Microsoft technologies. His focus has been primarily on .NET development since it was a beta product. He also enjoys teaching, skiing, hiking, hunting for wild mushrooms, and pool.

# Acknowledgments

**M**any people made this book possible. First and foremost, Jonathan Hassell and Elizabeth Seymour of Apress, Inc. drove the concept, writing, editing, and publication of the entire project. My corporate vice presidents at Volt Information Sciences, Inc., Patrick Walker and Christina Harris, suggested the idea of this book in the first place and supported its development. The lead technical reviewer, Josh Kelling (Kelling Consulting) did a terrific job at finding and correcting my coding mistakes. I'm also grateful to Doug Walter (Microsoft), who contributed significantly to the technical accuracy of this book. Many of the sections of this book are based on a monthly column I write for Microsoft's MSDN Magazine. My editors at MSDN, Joshua Trupin and Stephen Toub, provided me with a lot of advice about writing, without which this book would never have gotten off the ground. And finally, my staff at Volt—Shirley Lin, Lisa Vo Carlson, and Grace Son—supplied indispensable administrative help.

Many Volt software engineers working at Microsoft acted as auxiliary technical and editorial reviewers for this book. Primary technical reviewers include: Evan Kaplan, Steven Fusco, Bruce Ritter, Peter Yan, Ron Starr, Gordon Lippa, Kirk Slota, Joanna Tao, Walter Wittel, Jay Gray, Robert Hopkins, Sam Abolrous, Rich Bixby, Max Guernsey, Larry Briones, Kristin Jaeger, Joe Davis, Andrew Lee, Clint Kreider, Craig Green, Daniel Bedassa, Paul Kwiatkowski, Mark Wilcox, David Blais, Mustafa Al-Hasnawi, David Grossberg, Vladimir Abashyn, Mitchell Harter, Michael Svob, Brandon Lake, David Reynolds, Rob Gilmore, Cyrus Jamula, Ravichandhiran Kolandaiswamy, and Rajkumar Ramasamy.

Secondary technical reviewers include Jerry Frost, Michael Wansley, Vanarasi Antony Swamy, Ted Keith, Chad Fairbanks, Chris Trevino, David Moy, Fuhan Tian, C.J. Eichholz, Stuart Martin, Justice Chang, Funmi Bolonduro, Alemeshet Alemu, Lori Shih, Eric Mattoon, Luke Burtis, Aaron Rodriguez, Ajay Bhat, Carol Snyder, Qiusheng Gao, Haik Babaian, Jonathan Collins, Dinesh Ravva, Josh Silveria, Brian Miller, Gary Roehl, Kender Talyor, Ahlee Ly, Conan Callen, Kathy Davis, and Florentin Ionescu.

Editorial reviewers include Christina Zubelli, Joey Gonzales, Tony Chu, Alan Vandarwarka, Matt Carson, Tim Garner, Michael Klevitsky, Mark Soth, Michael Roshak, Robert Hawkins, Mark McGee, Grace Lou, Reza Sorasi, Abhijeet Shah, April McCready, Creede Lambard, Sean McCallum, Dawn Zhao, Mike Agranov, Victor Araya Cantuarias, Jason Olsan, Igor Bodi, Aldon Schwimmer, Andrea Borning, Norm Warren, Dale Dey, Chad Long, Thom Hokama, Ying Guo, Yong Wang, David Shockley, Allan Lockridge, Prashant Patil, Sunitha Mutnuri, Ping Du, Mark Camp, Abdul Khan, Moss Willow, Madhavi Kandibanda, John Mooney, Filiz Kurban, Jesse Larsen, Jeni Jordan, Chris Rosson, Dean Thomas, Brandon Barela, and Scott Lanphear.

# Introduction

## What This Book Is About

This book presents practical techniques for writing lightweight software test automation in a .NET environment. If you develop, test, or manage .NET software, you should find this book useful. Before .NET, writing test automation was often as difficult as writing the code for the application under test itself. With .NET, you can write lightweight, custom test automation in a fraction of the time it used to take. By *lightweight automation*, I mean small, dedicated test harness programs that are typically two pages of source code or less in length and take less than two hours to write. The emphasis of this book is on practical techniques that you can use immediately.

## Who This Book Is For

This book is intended for software developers, testers, and managers who work with .NET technology. This book assumes you have a basic familiarity with .NET programming but does not make any particular assumptions about your skill level. The examples in this book have been successfully used in seminars where the audience background has ranged from beginning application programmers to advanced systems programmers. The content in this book has also been used in teaching environments where it has proven highly effective as a platform for students who are learning intermediate level .NET programming.

## Advantages of Lightweight Test Automation

The automation techniques in this book are intended to complement, not replace, other testing paradigms, such as manual testing, test-driven development, model-based testing, open source test frameworks, commercial test frameworks, and so on. Software test automation, including the techniques in this book, has five advantages over manual testing. We sometimes refer to these automation advantages with the acronym SAPES: test automation has better Speed, Accuracy, Precision, Efficiency, and Skill-Building than manual testing. Additionally, when compared with both open source test frameworks and commercial frameworks, lightweight test automation has the advantage of not requiring you to travel up a rather steep learning curve and perhaps even learning a proprietary scripting language. Compared with commercial test automation frameworks, lightweight test automation is much less expensive and is fully customizable. And compared with open source test frameworks, lightweight automation is more stable in the sense that you have fewer recurring version updates and bug fixes to deal with. But the single most important advantage of lightweight, custom test automation harnesses over commercial and open source test frameworks is subjective—lightweight automation actively encourages and promotes creative testing, whereas commercial and open source frameworks often tend to direct the types of automation you create to the types of tests that are best supported by the framework. The single biggest disadvantage of lightweight test automation is manageability. Because lightweight test harnesses are so easy to write, if you

aren't careful, your testing effort can become overwhelmed by the sheer number of test harnesses, test case data, and test case result files you create. Test process management is outside the scope of this book, but it is a challenging topic you should not underestimate when writing lightweight test automation.

## Coding Issues

All the code in this book is written in the C# language. Because of the unifying influence of the underlying .NET Framework, you can refactor the code in this book to Visual Basic .NET without too much trouble if necessary. All the code in this book was tested and ran successfully on both Windows XP Professional (SP2) and Windows Server 2003, and with Visual Studio .NET 2003 (with Framework 1.1) and SQL Server 2000. The code was also tested on Visual Studio 2005 (with Framework 2.0) and SQL Server 2005; however, if you are developing in that environment, you'll have to make a few minor changes. I've coded the examples so that any changes you have to make for VS 2005 and SQL Server 2005 are flagged quickly. I decided that presenting just code for VS 2003 and SQL Server 2000 was a better approach than to sprinkle the book text with many short notes describing the minor development platform differences for VS 2005 and SQL Server 2005. The code in this book is intended strictly for 32-bit systems and has not been tested against 64-bit systems.

If you are new to software test automation, you'll quickly find that coding as a tester is significantly different from coding as a developer. Most of the techniques in this book are coded using a traditional, scripting style, rather than in an object-oriented style. I've found that automation code is easier to understand when written in a scripting style but this is a matter of opinion. Also, most of the code examples are not parameterized or packaged as methods. Again, this is for clarity. Most of the normal error-checking code, such as checking the values of input parameters to methods, is omitted. Error-traps are absolutely essential in production test automation code (after all, you are expecting to find errors) but error-checking code is often three or four times the size of the core code being checked. The code in this book is specifically designed for you to modify, which includes wrapping into methods, adding error-checks, incorporating into other test frameworks, and encapsulating into utility classes and libraries.

Most of the chapters in this book present dummy applications to test against. By design, these dummy applications are not examples of good coding style, and these applications under test often contain deliberate errors. This keeps the size of the dummy applications small and also simulates the unrefined nature of an application's state during the development process. For example, I generally use default control names such as `textBox1` rather than use descriptive names, I keep local variable names short (such as `s` for a string variable), I sometimes place multiple statements on the same line, and so forth. I've actually left a few minor "severity 4" bugs (typographical errors) in the screenshots in this book; you might enjoy looking for them.

In most cases, I've tried to be as accurate as possible with my terminology. For example, I use the term *method* when dealing with a subroutine that is a field/member in a C# class, and I use the term *function* when referring to a C++ subroutine in a Win32 API library. However, I make exceptions when I feel that a slightly incorrect term is more understandable or readable. For example, I sometimes use the term *string variable* instead of the more accurate *string object* when referring to a C# string type item.

This book uses a problem-solution structure. This approach has the advantage of organizing various test automation tasks in a convenient way. But to keep the size of the book reasonable, most of the solutions are not complete, standalone blocks of code. This means

that I often do not declare variables, explicitly discuss the namespaces and project references used in the solution, and so on. Many of the solutions in a chapter refer to other solutions within the same chapter, so you'll have to make reasonable assumptions about dependencies and how to turn the solution code into complete test harnesses. To assist you in understanding how the sections of a chapter work together, the last section of every chapter presents a complete, standalone program.

## Contents of This Book

In most computer science books, the contents of the book are summarized in the introduction. I will forego that practice and say instead that the best way to get a feel for what is contained in this book is to scan the table of contents; I know that's what I always do. That said however, let me mention four specific topics in this book that have generated particular interest among my colleagues. Chapter 1, "API Testing," is in many ways the most fundamental type of all software testing. If you are new to software testing, you will not only learn useful testing techniques, but you'll also learn many of the basic principles of software testing. Chapter 3, "Windows-Based UI Testing," presents powerful techniques to manipulate an application through its user interface. Even software testers with many years of experience are surprised at how easy UI test automation is using .NET and the techniques in that chapter. Chapter 5, "Request-Response Testing," demonstrates the basic techniques to test any Web-based application. Web developers and testers are frequently surprised at how powerful these techniques are in a .NET environment. Chapter 10, "Combinations and Permutations," gives you the tools you need to programmatically generate test cases that take into account all combinations and rearrangements of input values. Both new and experienced testers have commented that combinatorics with .NET makes test case generation significantly more efficient than previously.

## Using the Code in This Book

This book is intended to provide practical help for you in developing and testing software. This means that, within reason, you may use the code in this book in your systems and documentation. Obvious exceptions include situations where you are reproducing a significant portion of the code in this book on a Web site or magazine article, or using examples in a conference talk, and so on. Most authors, including me, appreciate citations if you use examples from their book in a paper or article. All code is provided without warranty of any kind.