

# Contents

Acknowledgement . . . . .	i
Foreword . . . . .	iii
About the author . . . . .	v
About this book . . . . .	vii
<b>1 Introduction to Parallel Extensions</b>	<b>1</b>
1.1 History of parallelization . . . . .	2
1.2 Parallelization and LINQ . . . . .	4
1.2.1 A quick introduction to LINQ . . . . .	4
1.2.2 Looking at the egg boiling problem . . . . .	6
1.2.3 Introducing LINQ to simplify and optimize . . . . .	10
1.2.4 Parallelizing it! . . . . .	11
1.3 Summary . . . . .	14
<b>2 Productivity and Quality with Unit Testing</b>	<b>15</b>
2.1 Avoid too many manual tests . . . . .	16
2.2 What is Test Driven Development? . . . . .	16
2.2.1 Make it fail . . . . .	17
2.2.2 Make it work . . . . .	18
2.2.3 Make it better . . . . .	18
2.3 Using MS Test . . . . .	19
2.3.1 Introducing more tests . . . . .	22
2.3.2 Alternatives to MS Test . . . . .	27
2.4 Summary . . . . .	28
<b>3 Is upgrading your code a productive step?</b>	<b>29</b>
3.1 What is considered an upgrade? . . . . .	30
3.2 Changing behavior on existing features . . . . .	31
3.3 Upgrading the code base . . . . .	32
3.4 When will you reach the final frontier? . . . . .	33
3.4.1 It is time to re-write the code . . . . .	33
3.5 Providing upgrades and hot-fixes continuously . . . . .	34
3.5.1 Working with ClickOnce . . . . .	34
3.5.2 Managing Online Application upgrades . . . . .	38

3.6	Being more time efficient . . . . .	39
3.7	Different tools that will help you along the way . . . . .	39
3.7.1	Being productive with ReSharper . . . . .	39
3.7.2	Analyzing code with NDepend . . . . .	49
3.7.3	Creating a report with Visual NDepend . . . . .	51
3.8	Summary . . . . .	60
<b>4</b>	<b>Creating a challenge out of the trivial tasks</b>	<b>61</b>
4.1	Improving your work . . . . .	62
4.2	Improving architecture over time . . . . .	66
4.2.1	Avoid slacking off . . . . .	67
4.3	Summary . . . . .	70
<b>5</b>	<b>Asynchronous programming with async and await</b>	<b>71</b>
5.1	Asynchronous vs Parallel . . . . .	72
5.2	How did we solve this before? . . . . .	74
5.2.1	Using the BackgroundWorker . . . . .	74
5.2.2	Using Task Parallel Library (TPL) . . . . .	77
5.3	Introducing a new pattern . . . . .	81
5.4	Refactoring a synchronous application . . . . .	86
5.5	Summary . . . . .	90
<b>6</b>	<b>Dynamic programming</b>	<b>91</b>
6.1	What is dynamic programming . . . . .	92
6.1.1	Introducing the ExpandableObject . . . . .	92
6.2	Going from Dynamic to More Dynamic . . . . .	94
6.3	Dynamic in the real world . . . . .	100
6.3.1	Introducing IronPython . . . . .	100
6.4	Summary . . . . .	109
<b>7</b>	<b>Increase readability with anonymous types and methods</b>	<b>111</b>
7.1	What is anonymity . . . . .	112
7.2	Digging into Anonymous types . . . . .	114
7.2.1	Delegates and Events . . . . .	114
7.2.2	Introducing anonymity . . . . .	115
7.2.3	Anonymity and readability . . . . .	121
7.3	Summary . . . . .	124
<b>8</b>	<b>Exploring Reflection</b>	<b>125</b>
8.1	Scanning objects for information . . . . .	126
8.1.1	Getting the type of the object . . . . .	126
8.1.2	Getting all the properties from an object . . . . .	127
8.1.3	Getting a value from a specific property . . . . .	127
8.2	Creating a generic search . . . . .	128
8.2.1	Implement the search method . . . . .	129

8.3	What is reflection and why is it useful? . . . . .	135
8.3.1	Getting information about methods . . . . .	136
8.3.2	Why reflection is so powerful . . . . .	140
8.4	Digging deeper with Reflection . . . . .	140
8.4.1	Setting values on objects . . . . .	140
8.4.2	Getting information about attributes . . . . .	149
8.5	Summary . . . . .	152
<b>9</b>	<b>Creating things at runtime</b>	<b>153</b>
9.1	What makes it so effective? . . . . .	154
9.2	Creating methods with DynamicMethod . . . . .	155
9.2.1	Defining the dynamic method . . . . .	156
9.2.2	Emitting IL . . . . .	156
9.2.3	Using the OpCodes . . . . .	158
9.2.4	Invoking the dynamically created method . . . . .	158
9.2.5	How Dynamic Methods affect resources . . . . .	162
9.3	Invoking things from a dynamic method . . . . .	167
9.3.1	Invoking another dynamic method . . . . .	170
9.4	Exploring Microsoft IL . . . . .	175
9.4.1	The evaluation stack . . . . .	175
9.4.2	Creating recursive dynamic methods . . . . .	181
9.4.3	Creating a Switch . . . . .	188
9.5	Summary . . . . .	195
<b>10</b>	<b>Introducing Roslyn</b>	<b>197</b>
10.1	Why is Roslyn important? . . . . .	198
10.2	Getting Roslyn running . . . . .	198
10.2.1	Running some code with Roslyn . . . . .	199
10.3	Extracting the information from a C# code file . . . . .	200
10.4	Analyzing code with Roslyn . . . . .	203
10.5	Using the C# Interactive Window . . . . .	209
10.5.1	Accessing types in your solution . . . . .	216
10.6	Summary . . . . .	219
<b>11</b>	<b>Adapting to Inversion of Control</b>	<b>221</b>
11.1	Understanding the principle . . . . .	222
11.2	Cleaning up the mess . . . . .	227
11.3	Introducing an IoC container . . . . .	230
11.4	Testable code . . . . .	233
11.5	Summary . . . . .	236
<b>12</b>	<b>Are you Mocking me?</b>	<b>237</b>
12.1	Where you might have seen Mocking before . . . . .	238
12.2	How does this apply to programming? . . . . .	238
12.3	Mocking with Simple.Mocking . . . . .	241

12.3.1 Let us start mocking! . . . . .	243
12.4 Summary . . . . .	256
<b>Index</b>	<b>257</b>