

Foundations of Object-Oriented Programming Using .NET 2.0 Patterns



Christian Gross

Foundations of Object-Oriented Programming Using .NET 2.0 Patterns

Copyright © 2006 by Christian Gross

Lead Editor: Jonathan Hassell

Technical Reviewer: Brian Myers

Editorial Board: Steve Anglin, Dan Appleman, Ewan Buckingham, Gary Cornell, Tony Davis, Jason Gilmore,

Jonathan Hassell, Chris Mills, Dominic Shakeshaft, Jim Sumser

Project Manager: Sofia Marchant

Copy Edit Manager: Nicole LeClerc

Copy Editor: Ami Knox

Assistant Production Director: Kari Brooks-Copony

Production Editor: Linda Marousek

Compositor: Susan Glinert Stevens

Proofreader: Elizabeth Berry

Indexer: Valerie Perry

Interior Designer: Van Winkle Design Group

Cover Designer: Kurt Krames

Manufacturing Director: Tom Debolski

Library of Congress Cataloging-in-Publication Data

Gross, Christian.

Foundations of object-oriented programming using .NET 2.0 patterns / Christian Gross.

p. cm.

ISBN 1-59059-540-8

1. Object-oriented programming (Computer science) 2. Microsoft .NET. 3. Software patterns. I. Title.

QA76.64.G8 2005

005.1'17--dc22

2005025961

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.

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.

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, 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, 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. You will need to answer questions pertaining to this book in order to successfully download the code.

Contents

About the Author	xi
About the Technical Reviewer	xiii
Introduction	xv

■ CHAPTER 1 Essentials of Object-Oriented Programming 1

Understanding Types	1
Modular vs. Reusable Code	3
Using Scope to Control Access	6
Understanding Inheritance	7
Differences Between Inheritance and Structs	8
Simple Inheritance	12
Inheritance Using Virtual Functions	13
Inheritance in a Software-Engineering Context	14
Writing Generic Code	16
The Case for Generics	16
Constraints	17
Some Considerations	19
Composition	19
Asynchronous Code	20
Some Final Thoughts	22

■ CHAPTER 2 Logging, Errors, and Test-Driven Development 23

Implementing Logging Management	23
A Simple Log4net Example	23
Generating Messages Using the ILog Interface	26
Managing a Configuration File	27
Setting Up a Real-Life Configuration	29
Implementing ToString	36
Implementing Exception Handling	37
Classifying an Error and an Exception	37
Implementing an Error Strategy	40
Implementing an Exception Strategy	47

Using NUnit for Test-Driven Development	52
Understanding Test-Driven Development	52
Enabling NUnit in Your Application	53
Running NUnit Tests	54
More Detailed NUnit Tests	55
Some Final Thoughts	58
 CHAPTER 3 Defining the Foundation	61
Defining a Base for the Application	61
Defining Intentions	61
Implementing Test-Driven Development	63
Implementing the Bridge Pattern	68
Keeping Options Open Using Interfaces	69
Too Much of a Good Thing	69
What About .NET 1.x?	70
Bridge Pattern Implementation Variations	71
Implementing Application Logic	71
Controller Interfaces	75
Implementing a Default Base Class	76
Interface and Class Design Decisions	80
Instantiating Types with the Factory Pattern	84
The Need for a Helper Type	84
Implementing a Plug-In Architecture	85
Creating Objects According to a Plan	86
Cloning Objects	87
Some Final Thoughts	88
 CHAPTER 4 Application Architecture	89
Making an Application Work Properly	89
Extensibility and Maintainability	90
Using Black Boxes	90
Pipes and Filters Pattern	92
An Example: Buying a Television from Amazon.com	93
Architecting the Television Selection System	94
Implementing the Television Selection System	95
Some Final Notes About the Pipes and Filters Pattern	104

Client-Dispatcher-Server Pattern	105
Defining the Client-Dispatcher-Server Pattern Architecture	106
A Static Dispatcher Architecture	106
A Dynamic Dispatcher Architecture	108
Architecting the Client-Dispatcher-Server Pattern	110
Implementing the Assembly Directory Resolver	113
Implementing a Web Service Resolver	118
Micro-Kernel Pattern	120
Architecture of the Micro-Kernel	121
Hiding the Details of the Micro-Kernel	122
Designing a Micro-Kernel	123
Micro-Kernel Implementation Details	124
Building a Simple Banking Application	125
Some Final Considerations for the Micro-Kernel Pattern	130
Some Final Thoughts	131

■ CHAPTER 5 **Implementing Component Groupings** 133

Two Traditional Object-Oriented Faux Pas	133
Properties and Controlling an Oven's Temperature	134
Inheritance and the Fragile Base Class Problem	136
Sample Application: A Translation Program	141
Writing a Quick-and-Dirty Application	141
Refactoring Code	143
Refactoring and Implementing a Bridge and Factory	144
Implementing the Mediator Pattern	145
Implementing the Template Pattern	151
Implementing the Adapter Pattern	156
Some Final Notes About the Application	160
Adding Multiple Languages to the Application	161
Look Up: Is It a Decorator, Is It a Composite?	161
Implementing the Chain of Responsibility Pattern	162
Implementing the Command Pattern	167
Implementing the Composite Pattern	170
Implementing the Decorator Pattern	173
Implementing the State Pattern	176
Implementing the Strategy Pattern	180
Implementing Dynamic Selection of Language Translation	182
Some Final Thoughts	184

CHAPTER 6	Writing Algorithms	185
	Impersonating Functionality Without Modifications	185
	Implementing the Proxy Pattern	186
	Enhancing Types Using Functors	188
	Creating a Generic Functor Architecture for Collections	197
	Building a Movie Ticket Application	200
	Starting with the Basics	200
	Calculating Ticket Sales	201
	Reading Ticket Sales Data	202
	The Problem of Using Null	205
	A Simpler Way to Buy Tickets: Using the Façade Pattern	208
	Managing Extensions Using Polymorphism	212
	Implementing the Static Extension Pattern	212
	Implementing the Dynamic Extension Pattern	217
	Extensions, Typecasting, and What It All Means	222
	Looping Through Your Data Using the Iterator Pattern	223
	Implementing the Iterator Pattern Using C# 2.0	223
	Using Functors in an Iterator	224
	Some Final Thoughts	226
 CHAPTER 7	 Efficient Code	 227
	Immutable Classes Are Efficient Classes	227
	Why Immutable Classes Are Consistent	228
	Why Immutable Class Are Scalable	234
	Some Rules of Thumb for Immutable Classes	238
	Using Immutable Classes in the Flyweight Pattern	239
	An Example of the Flyweight Pattern	240
	A Generic Flyweight Architecture	241
	Using the Generic Flyweight Architecture	243
	Using the Flyweight Implementation	244
	The Theory of Object Pools	246
	Object Pools and COM+	246
	The Theory of Object Pools	247
	Implementing an Object Pool Pattern in .NET	247
	Multithreaded Applications	254
	A Simple Thread Example	255
	Implementing the Singleton	256
	Managing Multithreaded Problems Using the Producer-Consumer Technique	266
	Some Final Thoughts	269

CHAPTER 8	Data Persistence	271
	Serialization in .NET	272
	Binary Object Serialization in .NET	273
	XML Object Serialization in .NET	274
	Serialization Has Issues!	276
	Tweaking and Fixing the Serializer Pattern	277
	Accessing an External State: The Visitor Pattern	278
	Accessing an Internal State: The Memento Pattern	286
	Object-to-Relational Data Mapping Using NHibernate	292
	A Simple NHibernate Example	293
	Mapping One-to-Many Relationships	301
	Other Types of Relationships	310
	Using HQL	310
	Some Final Thoughts	312
CHAPTER 9	Refactoring to Patterns	313
	Test-Driven Development and Refactoring	313
	Writing That First Line of Code	314
	After That First Line of Code	315
	Kinds of Refactorings	315
	Classes, Methods—Everything Is Too Big	317
	Refactoring the Stream Class	319
	Problems of Refactoring the Stream Class	321
	Refactoring a Class and Not a Base Type	322
	I Can't Understand the Code	325
	Approaching Unknown Code	325
	Tracer Code	326
	Breaking Some Code	326
	The Code Feels Similar, Yet Is Different	327
	Why Copying and Pasting Code Works	327
	Refactoring Duplicated Code Using the Template Method	328
	Duplication That Is Acceptable	333
	Time Is Short, Make It Quick	333
	I Wish This Code Was Removed	334
	Some Final Thoughts	335
INDEX		337