

THE ADDISON-WESLEY MICROSOFT TECHNOLOGY SERIES



THIRD EDITION

CONVENTIONS, IDIOMS, AND
PATTERNS FOR REUSABLE .NET LIBRARIES

FRAMEWORK DESIGN GUIDELINES

KRZYSZTOF C WALINA
JEREMY BARTON
BRAD ABRAMS

Forewords by SCOTT GUTHRIE,
MIGUEL DE ICAZA, and ANDERS HEJLSBERG



Framework Design Guidelines

Third Edition

Framework Design Guidelines: Conventions, Idioms, and Patterns for Reusable .NET Libraries

Table of Contents

Cover	
Half Title	
Title page	
Copyright Page	
Dedication	
Contents	
Figures	
Tables	
Foreword	
Foreword to the Second Edition	
Foreword to the First Edition	
Preface	
Acknowledgments	
About the Authors	
About the Annotators	
1 Introduction	
1.1 Qualities of a Well-Designed Framework	
1.1.1 Well-Designed Frameworks Are Simple	

Table of Contents

- 1.1.2 Well-Designed Frameworks Are Expensive to Design
- 1.1.3 Well-Designed Frameworks Are Full of Trade-Offs
- 1.1.4 Well-Designed Frameworks Borrow from the Past
- 1.1.5 Well-Designed Frameworks Are Designed to Evolve
- 1.1.6 Well-Designed Frameworks Are Integrated
- 1.1.7 Well-Designed Frameworks Are Consistent

2 Framework Design Fundamentals

2.1 Progressive Frameworks

2.2 Fundamental Principles of Framework Design

- 2.2.1 The Principle of Scenario-Driven Design
- 2.2.2 The Principle of Low Barrier to Entry
- 2.2.3 The Principle of Self-Documenting Object Models
- 2.2.4 The Principle of Layered Architecture

3 Naming Guidelines

3.1 Capitalization Conventions

- 3.1.1 Capitalization Rules for Identifiers
- 3.1.2 Capitalizing Acronyms
- 3.1.3 Capitalizing Compound Words and Common Terms
- 3.1.4 Case Sensitivity

3.2 General Naming Conventions

- 3.2.1 Word Choice
- 3.2.2 Using Abbreviations and Acronyms
- 3.2.3 Avoiding Language-Specific Names
- 3.2.4 Naming New Versions of Existing APIs

3.3 Names of Assemblies, DLLs, and Packages

3.4 Names of Namespaces

- 3.4.1 Namespaces and Type Name Conflicts

Table of Contents

3.5 Names of Classes, Structs, and Interfaces

3.5.1 Names of Generic Type Parameters

3.5.2 Names of Common Types

3.5.3 Naming Enumerations

3.6 Names of Type Members

3.6.1 Names of Methods

3.6.2 Names of Properties

3.6.3 Names of Events

3.6.4 Naming Fields

3.7 Naming Parameters

3.7.1 Naming Operator Overload Parameters

3.8 Naming Resources

4 Type Design Guidelines

4.1 Types and Namespaces

4.2 Choosing Between Class and Struct

4.3 Choosing Between Class and Interface

4.4 Abstract Class Design

4.5 Static Class Design

4.6 Interface Design

4.7 Struct Design

4.8 Enum Design

4.8.1 Designing Flag Enums

4.8.2 Adding Values to Enums

4.9 Nested Types

4.10 Types and Assembly Metadata

4.11 Strongly Typed Strings

Table of Contents

5 Member Design

5.1 General Member Design Guidelines

- 5.1.1 Member Overloading
- 5.1.2 Implementing Interface Members Explicitly
- 5.1.3 Choosing Between Properties and Methods

5.2 Property Design

- 5.2.1 Indexed Property Design
- 5.2.2 Property Change Notification Events

5.3 Constructor Design

- 5.3.1 Type Constructor Guidelines

5.4 Event Design

5.5 Field Design

5.6 Extension Methods

5.7 Operator Overloads

- 5.7.1 Overloading Operator
- 5.7.2 Conversion Operators
- 5.7.3 Inequality Operators

5.8 Parameter Design

- 5.8.1 Choosing Between Enum and Boolean Parameters
- 5.8.2 Validating Arguments
- 5.8.3 Parameter Passing
- 5.8.4 Members with Variable Number of Parameters
- 5.8.5 Pointer Parameters

5.9 Using Tuples in Member Signatures

6 Designing for Extensibility

6.1 Extensibility Mechanisms

- 6.1.1 Unsealed Classes

Table of Contents

6.1.2 Protected Members

6.1.3 Events and Callbacks

6.1.4 Virtual Members

6.1.5 Abstractions (Abstract Types and Interfaces)

6.2 Base Classes

6.3 Sealing

7 Exceptions

7.1 Exception Throwing

7.2 Choosing the Right Type of Exception to Throw

7.2.1 Error Message Design

7.2.2 Exception Handling

7.2.3 Wrapping Exceptions

7.3 Using Standard Exception Types

7.3.1 Exception and SystemException

7.3.2 ApplicationException

7.3.3 InvalidOperationException

7.3.4 ArgumentException, ArgumentNullException, and
ArgumentOutOfRangeException

7.3.5 NullReferenceException, IndexOutOfRangeException, and
AccessViolationException

7.3.6 StackOverflowException

7.3.7 OutOfMemoryException

7.3.8 ComException, SEHException, and ExecutionEngineException

7.3.9 OperationCanceledException and TaskCanceledException

7.3.10 FormatException

7.3.11 PlatformNotSupportedException

7.4 Designing Custom Exceptions

7.5 Exceptions and Performance

Table of Contents

7.5.1 The TestDoer Pattern

7.5.2 The Try Pattern

8 Usage Guidelines

8.1 Arrays

8.2 Attributes

8.3 Collections

8.3.1 Collection Parameters

8.3.2 Collection Properties and Return Values

8.3.3 Choosing Between Arrays and Collections

8.3.4 Implementing Custom Collections

8.4 DateTime and DateTimeOffset

8.5 ICloneable

8.6 IComparable<T> and IEquatable<T>

8.7 IDisposable

8.8 Nullable<T>

8.9 Object

8.9.1 Object.Equals

8.9.2 Object.GetHashCode

8.9.3 Object.ToString

8.10 Serialization

8.11 Uri

8.11.1 System.Uri Implementation Guidelines

8.12 System.Xml Usage

8.13 Equality Operators

8.13.1 Equality Operators on Value Types

8.13.2 Equality Operators on Reference Types

9 Common Design Patterns

Table of Contents

9.1 Aggregate Components

- 9.1.1 Component-Oriented Design
- 9.1.2 Factored Types
- 9.1.3 Aggregate Component Guidelines

9.2 The Async Patterns

- 9.2.1 Choosing Between the Async Patterns
- 9.2.2 Task-Based Async Pattern
- 9.2.3 Async Method Return Types
- 9.2.4 Making an Async Variant of an Existing Synchronous Method
- 9.2.5 Implementation Guidelines for Async Pattern Consistency
- 9.2.7 Classic Async Pattern
- 9.2.8 Event-Based Async Pattern
- 9.2.9 `IAsyncDisposable`
- 9.2.10 `IAsyncEnumerable<T>`

9.3 Dependency Properties

- 9.3.1 Dependency Property Design
- 9.3.2 Attached Dependency Property Design
- 9.3.3 Dependency Property Validation
- 9.3.4 Dependency Property Change Notifications
- 9.3.5 Dependency Property Value Coercion

9.4 Dispose Pattern

- 9.4.1 Basic Dispose Pattern
- 9.4.2 Finalizable Types
- 9.4.3 Scoped Operations
- 9.4.4 `IAsyncDisposable`

9.5 Factories

9.6 LINQ Support

- 9.6.1 Overview of LINQ

Table of Contents

- 9.6.2 Ways of Implementing LINQ Support
- 9.6.3 Supporting LINQ through IEnumerable<T>
- 9.6.4 Supporting LINQ through IQueryable<T>
- 9.6.5 Supporting LINQ through the Query Pattern

9.7 Optional Feature Pattern

9.8 Covariance and Contravariance

- 9.8.1 Contravariance
- 9.8.2 Covariance
- 9.8.3 Simulated Covariance Pattern

9.9 Template Method

9.10 Timeouts

9.11 XAML Readable Types

9.12 Operating on Buffers

- 9.12.1 Data Transformation Operations
- 9.12.2 Writing Fixed or Predetermined Sizes to a Buffer
- 9.12.3 Writing Values to Buffers with the Try-Write Pattern
- 9.12.4 Partial Writes to Buffers and OperationStatus

9.13 And in the End

A: C# Coding Style Conventions

A.1 General Style Conventions

- A.1.1 Brace Usage
- A.1.2 Space Usage
- A.1.3 Indent Usage
- A.1.4 Vertical Whitespace
- A.1.5 Member Modifiers
- A.1.6 Other

A.2 Naming Conventions

Table of Contents

A.3 Comments

A.4 File Organization

B: Obsolete Guidance

B.3 Obsolete Guidance from Naming Guidelines

B.3.8 Naming Resources

B.4 Obsolete Guidance from Type Design Guidelines

B.4.1 Types and Namespaces

B.5 Obsolete Guidance from Member Design

B.5.4 Event Design

B.7 Obsolete Guidance from Exceptions

B.7.4 Designing Custom Exceptions

B.8 Obsolete Guidance from Usage Guidelines

B.8.10 Serialization

B.9 Obsolete Guidance from Common Design Patterns

B.9.2 The Async Patterns

B.9.4 Dispose Pattern

C: Sample API Specification

D: Breaking Changes

D.1 Modifying Assemblies

D.1.1 Changing the Name of an Assembly

D.2 Adding Namespaces

D.2.1 Adding a Namespace That Conflicts with an Existing Type

D.3 Modifying Namespaces

D.3.1 Changing the Name or Casing of a Namespace

D.4 Moving Types

D.4.1 Moving a Type via [TypeForwardedTo]

D.4.2 Moving a Type Without [TypeForwardedTo]

Table of Contents

D.5 Removing Types

D.5.1 Removing Types

D.6 Modifying Types

D.6.1 Sealing an Unsealed Type

D.6.2 Unsealing a Sealed Type

D.6.3 Changing the Case of a Type Name

D.6.4 Changing a Type Name

D.6.5 Changing the Namespace for a Type

D.6.6 Adding readonly on a struct

D.6.7 Removing readonly from a struct

D.6.8 Adding a Base Interface to an Existing Interface

D.6.9 Adding the Second Declaration of a Generic Interface

D.6.10 Changing a class to a struct

D.6.11 Changing a struct to a class

D.6.12 Changing a struct to a ref struct

D.6.13 Changing a ref struct to a (Non-ref) struct

D.7 Adding Members

D.7.1 Masking Base Members with new

D.7.2 Adding abstract Members

D.7.3 Adding Members to an Unsealed Type

D.7.4 Adding an override Member to an Unsealed Type

D.7.5 Adding the First Reference Type Field to a struct

D.7.6 Adding a Member to an Interface

D.8 Moving Members

D.8.1 Moving Members to a Base Class

D.8.2 Moving Members to a Base Interface

D.8.3 Moving Members to a Derived Type

D.9 Removing Members

Table of Contents

D.9.1 Removing a Finalizer from an Unsealed Type

D.9.2 Removing a Finalizer from a Sealed Type

D.9.3 Removing a Non-override Member

D.9.4 Removing an override of a virtual Member

D.9.5 Removing an override of an abstract Member

D.9.6 Removing or Renaming Private Fields on Serializable Types

D.10 Overloading Members

D.10.1 Adding the First Overload of a Member

D.10.2 Adding Alternative-Parameter Overloads for a Reference
Type Parameter

D.11 Changing Member Signatures

D.11.1 Renaming a Method Parameter

D.11.2 Adding or Removing a Method Parameter

D.11.3 Changing a Method Parameter Type

D.11.4 Reordering Method Parameters of Differing Types

D.11.5 Reordering Method Parameters of The Same Type

D.11.6 Changing a Method Return Type

D.11.7 Changing the Type of a Property

D.11.8 Changing Member Visibility from public to Any Other Visibility

D.11.9 Changing Member Visibility from protected to public

D.11.10 Changing a virtual (or abstract) Member from protected to public

D.11.11 Adding or Removing the static Modifier

D.11.12 Changing to or from Passing a Parameter by Reference

D.11.13 Changing By-Reference Parameter Styles

D.11.14 Adding the readonly Modifier to a struct Method

D.11.15 Removing the readonly Modifier from a struct Method

D.11.16 Changing a Parameter from Required to Optional

D.11.17 Changing a Parameter from Optional to Required

D.11.18 Changing the Default Value for an Optional Parameter

Table of Contents

D.11.19 Changing the Value of a const Field

D.11.20 Changing an abstract Member to virtual

D.11.21 Changing a virtual Member to abstract

D.11.22 Changing a Non-virtual Member to virtual

D.12 Changing Behavior

D.12.1 Changing Runtime Error Exceptions to Usage Error Exceptions

D.12.2 Changing Usage Error Exceptions to Functioning Behavior

D.12.3 Changing the Type of Values Returned from a Method

D.12.4 Throwing a New Type of Error Exception

D.12.5 Throwing a New Type of Exception, Derived from an Existing
Thrown Type

D.13 A Final Note

Glossary

A

B

C

D

E

F

G

H

I

J

L

M

N

O

Table of Contents

P

R

S

T

U

V

X

Index