

Инфраструктура программных проектов

Соглашения, идиомы и шаблоны для
многократно используемых библиотек .NET

Framework Design Guidelines

Conventions, Idioms, and Patterns
For Reusable .Net Libraries

Second Edition

Crzysztof Cwalina
Brad Abrams



ADDISON-WESLEY

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokio • Singapore • Mexico City

Инфраструктура программных проектов

Соглашения, идиомы и шаблоны для
многократно используемых библиотек .NET

Второе издание

**Кржиштоф Цвалина
Брэд Абрамс**



Москва • Санкт-Петербург • Киев
2011

ББК 32.973.26-018.2.75

Ц25

УДК 681.3.07

Издательский дом "Вильямс"
Зав. редакцией С.Н. Тригуб
Перевод с английского и редакция Я.К. Шмидского

По общим вопросам обращайтесь в Издательский дом "Вильямс" по адресу:
info@williamspublishing.com, http://www.williamspublishing.com

Цвалина, Кржиштоф, Абрамс, Брэд.

Ц25 Инфраструктура программных проектов: соглашения, идиомы и шаблоны для многократно используемых библиотек .NET. : Пер. с англ. — М. : ООО "И.Д. Вильямс", 2011. — 416 с. : ил. — Парал. тит. англ.

ISBN 978-5-8459-1692-1 (рус.)

ББК 32.973.26-018.2.75

Все названия программных продуктов являются зарегистрированными торговыми марками соответствующих фирм.

Никакая часть настоящего издания ни в каких целях не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами, будь то электронные или механические, включая фотокопирование и запись на магнитный носитель, если на это нет письменного разрешения издательства PEARSON EDUCATION.

Authorized translation from the English language edition published by Addison-Wesley Publishing Company, Inc, Copyright © 2009 Microsoft Corporation

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher.

Russian language edition published by Williams Publishing House according to the Agreement with R&I Enterprises International, Copyright © 2011

Научно-популярное издание

Кржиштоф Цвалина, Брэд Абрамс

**ИНФРАСТРУКТУРА ПРОГРАММНЫХ ПРОЕКТОВ:
СОГЛАШЕНИЯ, ИДИОМЫ И ШАБЛОНЫ
ДЛЯ МНОГОКРАТНО ИСПОЛЬЗУЕМЫХ БИБЛИОТЕК .NET**

Литературный редактор *Е.Д. Давидян*
Верстка *Л.В. Чернокозинская*
Художественный редактор *В.Г. Павлютин*
Корректор *Л.А. Гордиенко*

Подписано в печать 29.10.2010. Формат 70x100/16.

Гарнитура Times. Печать офсетная.

Усл. печ. л. 33,54. Уч.-изд. л. 23,37.

Тираж 1000 экз. Заказ № 0000.

Отпечатано по технологии StP
в ОАО "Печатный двор" им. А. М. Горького
197110, Санкт-Петербург, Чкаловский пр., 15.

ООО "И. Д. Вильямс", 127055, г. Москва, ул. Лесная, д. 43, стр. 1

ISBN 978-5-8459-1692-1 (рус.)

ISBN 978-0-321-54561-9 (англ.)

© Издательский дом "Вильямс", 2011

© Microsoft Corporation, 2009



Оглавление

1	Введение	37
2	Основные принципы разработки инфраструктуры	43
3	Рекомендации по обозначениям	69
4	Рекомендации по разработке типов	103
5	Проектирование членов	141
6	Проектирование с целью расширяемости	201
7	Исключения	217
8	Рекомендации по использованию	247
9	Общие шаблоны проектирования	285
A	Соглашения о стиле кодирования в C#	347
Б	Использование FxCop для проверки рекомендаций по разработке инфраструктур	355
В	Пример спецификации API	385
	Глоссарий	393
	Список рекомендуемой литературы	399
	Предметный указатель	403



Содержание

<i>Предисловие</i>	17
<i>Предисловие к первому изданию</i>	19
<i>Предисловие</i>	21
<i>Благодарности</i>	25
<i>Об авторах</i>	27
<i>О комментаторах</i>	29
1 Введение	37
1.1. Характеристики правильно спроектированной инфраструктуры	39
1.1.1. Правильно спроектированные инфраструктуры просты	39
1.1.2. Правильно спроектированные инфраструктуры обходятся дорого	40
1.1.3. Правильно спроектированные инфраструктуры полны компромиссов	40
1.1.4. Учет опыта в правильно спроектированных инфраструктурах	41
1.1.5. Правильно спроектированные инфраструктуры проектируются так, чтобы их можно было расширить	41
1.1.6. Правильно спроектированные инфраструктуры интегрированы	42
1.1.7. Правильно спроектированные инфраструктуры непротиворечивы	42
2 Основные принципы разработки инфраструктуры	43
2.1. Прогрессивные инфраструктуры	45
2.2. Фундаментальные принципы разработки инфраструктуры	48
2.2.1. Принцип дизайна, основанного на сценариях	49
2.2.2. Принцип низкого барьера для доступа	54
2.2.3. Принцип самодокументирования объектных моделей	59
2.2.4. Принцип многоуровневой архитектуры	65

3	Рекомендации по обозначениям	69
3.1.	Соглашения о прописных буквах	70
3.1.1.	Правила использования прописных букв в идентификаторах	70
3.1.2.	Набор акронимов прописными буквами	72
3.1.3.	Использование прописных букв в составных словах и общепринятых терминах	74
3.1.4.	Чувствительность к регистру	75
3.2.	Общие соглашения об именах	76
3.2.1.	Выбор слов	77
3.2.2.	Использование сокращений и акронимов	79
3.2.3.	Избегайте названий, специфических для языка	79
3.2.4.	Обозначение новых версий для существующих API	81
3.3.	Названия сборок и DLL	83
3.4.	Названия пространств имен	85
3.4.1.	Пространства имен и конфликты с именами типов	86
3.5.	Названия классов, структур и интерфейсов	88
3.5.1.	Названия параметров универсальных типов	91
3.5.2.	Названия общих типов	92
3.5.3.	Обозначение перечислений	93
3.6.	Имена членов типа	95
3.6.1.	Названия методов	95
3.6.2.	Названия свойств	95
3.6.3.	Названия событий	97
3.6.4.	Обозначение полей	98
3.7.	Обозначение параметров	99
3.7.1.	Обозначение параметров перегружаемых операторов	100
3.8.	Обозначение ресурсов	100
4	Рекомендации по разработке типов	103
4.1.	Типы и пространства имен	105
4.1.1.	Стандартные названия подпространств имен	108
4.2.	Выбор между классом и структурой	110
4.3.	Выбор между классом и интерфейсом	113
4.4.	Дизайн абстрактных классов	119
4.5.	Разработка статических классов	121
4.6.	Проект интерфейса	122
4.7.	Разработка структур	124

8 Оглавление

4.8. Разработка перечислений	126
4.8.1. Проектирование перечислений флажков	132
4.8.2. Добавление значений к перечислениям	136
4.9. Вложенные типы	137
4.10. Типы и метаданные сборки	139
5 Проектирование членов	141
5.1. Общие рекомендации по разработке членов	141
5.1.1. Перегрузка членов	141
5.1.2. Явная реализация членов интерфейса	147
5.1.3. Выбор между свойствами и методами	151
5.2. Разработка свойств	156
5.2.1. Разработка индексированных свойств	158
5.2.2. События, уведомляющие об изменении свойств	159
5.3. Разработка конструкторов	161
5.3.1. Рекомендации для конструктора типов	166
5.4. Разработка событий	168
5.4.1. Разработка обработчика событий пользователем	173
5.5. Разработка полей	174
5.6. Методы расширения	176
5.7. Перегрузка операторов	181
5.7.1. Перегрузка оператора <code>operator==</code>	185
5.7.2. Операторы преобразования	185
5.8. Разработка параметров	187
5.8.1. Выбор между перечислениями и булевыми параметрами	189
5.8.2. Проверка правильности параметров	190
5.8.3. Передача параметров	193
5.8.4. Члены с переменным числом параметров	196
5.8.5. Параметры-указатели	199
6 Проектирование с целью расширяемости	201
6.1. Механизмы расширяемости	201
6.1.1. Негерметичные классы	202
6.1.2. Защищенные члены	203
6.1.3. События и обратные вызовы	204
6.1.4. Виртуальные члены	208
6.1.5. Абстракции (абстрактные типы и интерфейсы)	210

6.2. Базовые классы	212
6.3. Герметизация	214
7 Исключения	217
7.1. Вызов исключения	221
7.2. Выбор правильного типа вызываемого исключения	226
7.2.1. Разработка сообщений об ошибках	229
7.2.2. Обработка особых ситуаций	231
7.2.3. Обертывание исключений	235
7.3. Использование стандартных типов исключений	237
7.3.1. Исключения <code>Exception</code> и <code>SystemException</code>	237
7.3.2. Исключение <code>ApplicationException</code>	237
7.3.3. Исключение <code>InvalidOperationException</code>	238
7.3.4. Исключения <code>ArgumentException</code> , <code>ArgumentNullException</code> и <code>ArgumentOutOfRangeException</code>	238
7.3.5. Исключения <code>NullReferenceException</code> , <code>IndexOutOfRangeException</code> и <code>AccessViolationException</code>	239
7.3.6. Исключение <code>StackOverflowException</code>	240
7.3.7. Исключение <code>OutOfMemoryException</code>	240
7.3.8. Исключения <code>ComException</code> , <code>SEHException</code> и <code>ExecutionEngineException</code>	241
7.4. Проектирование пользовательских исключений	241
7.5. Исключения и производительность	243
7.5.1. Шаблон <code>Tester-Doer</code>)	243
7.5.2. Шаблон <code>Try-Parse</code>)	244
8 Рекомендации по использованию	247
8.1. Массивы	247
8.2. Атрибуты	249
8.3. Коллекции	251
8.3.1. Параметры-коллекции	253
8.3.2. Свойства коллекции и возвращаемые значения	254
8.3.3. Выбор между массивами и коллекциями	257
8.3.4. Реализация пользовательских коллекций	258
8.4. Типы <code>DateTime</code> и <code>DateTimeOffset</code>	260
8.5. Интерфейс <code>ICloneable</code>	262
8.6. <code>IComparable<T></code> и <code>IComparable<T></code>	263

10 Оглавление

8.7. IDisposable	265
8.8. Nullable<T>	265
8.9. Объект Object	266
8.9.1. Object.Equals	266
8.9.2. Хеш-функция Object.GetHashCode	268
8.9.3. Метод Object.ToString	269
8.10. Сериализация — преобразование в последовательную форму	271
8.10.1. Выбор подходящей для поддержки технологии преобразования в последовательную форму	272
8.10.2. Поддержка преобразования в последовательную форму в соответствии с контрактом данных	273
8.10.3. Поддержка преобразования в последовательную XML-форму	277
8.10.4. Поддержка преобразования в последовательную форму во время выполнения	278
8.11. Тип Uri	280
8.12. Использование System.Xml	281
8.13. Операторы равенства	282
8.13.1. Операторы равенства на типах значений	283
8.13.2. Операторы равенства на ссылочных типах	283
9 Общие шаблоны проектирования	285
9.1. Составные компоненты	285
9.1.1. Компонентно-ориентированный дизайн	287
9.1.2. Факторизованные типы	289
9.1.3. Рекомендации по объединению составляющих	290
9.2. Асинхронные шаблоны Async	293
9.2.1. Выбор одного из асинхронных шаблонов Async	293
9.2.2. Классический асинхронный шаблон Classic Async	295
9.2.3. Пример базовой реализации классического асинхронного шаблона Classic Async	298
9.2.4. Основанный на событиях асинхронный шаблон Event-Based Async	299
9.2.5. Поддержка out- и ref-параметров	301
9.2.6. Поддержка отмены	301
9.2.7. Поддержка сообщения о продвижении	302
9.2.8. Поддержка приращений — возрастающих результатов	304
9.3. Зависимые свойства	304
9.3.1. Разработка зависимых свойств	305
9.3.2. Разработка прикрепленных зависимых свойств	307

9.3.3. Проверка правильности зависимого свойства	308
9.3.4. Уведомления об изменении зависимого свойства	309
9.3.5. Приведение типа данных значения зависимого свойства	310
9.4. Шаблон освобождения Dispose	311
9.4.1. Основной шаблон освобождения Basic Dispose	313
9.4.2. Финализируемые типы	318
9.5. Фабрики	322
9.6. Поддержка LINQ	325
9.6.1. Краткий обзор LINQ	326
9.6.2. Способы реализации поддержки LINQ	327
9.6.3. Поддержка LINQ через IEnumerable<T>	327
9.6.4. Поддержка LINQ через IQueryable<T>	328
9.6.5. Поддержка LINQ через шаблон запроса Query	329
9.7. Шаблон дополнительных возможностей Optional Feature	331
9.8. Моделирование ковариации	335
9.9. Шаблон Template Method	340
9.10. Тайм-аут (истечение времени ожидания)	341
9.11. Читаемые XAML-типы	343
9.12. И в заключение...	345
А Соглашения о стиле кодирования в C#	347
A.1. Общие соглашения о стиле	348
A.1.1. Использование скобок	348
A.1.2. Использование пробелов	349
A.1.3. Использование отступов	350
A.1.4. Другое	350
A.2. Соглашения об именах	351
A.3. Комментарии	351
A.4. Организация файлов	352
Б Использование FxCop для проверки рекомендаций по разработке инфраструктур	355
Б.1. Что такое FxCop	355
Б.2. Развитие FxCop	356
Б.3. Как это работает	357

12 ■ Оглавление

Б.4. Описание руководства по FxCop	357
Б.4.1. Правила FxCop для рекомендаций по обозначениям	357
Б.4.2. Правила FxCop для разработки типа	366
Б.4.3. Правила FxCop по разработке членов	369
Б.4.4. Правила FxCop для проектирования расширяемости	375
Б.4.5. Правила FxCop для исключений	375
Б.4.6. Правила FxCop по использованию инфраструктуры	378
Б.4.7. Правила FxCop для шаблонов разработки Design Patterns	382
В Пример спецификации API	385
1. Требования	
2. Спецификации API	
2.1. Сценарии	
2.2. Разработка API	
3. Функциональные спецификации	388
Глоссарий	393
Список рекомендуемой литературы	399
Предметный указатель	403

Отзывы о книге *Инфраструктура программных проектов*

“Эта книга — одна из тех книг, которую на различных уровнях могут читать разработчики самых разнообразных программных продуктов. Независимо от того, проектируете вы эффективную объектную модель или хотите лучше понять .NET Framework, заимствовать опыт программистов-гуру, избежать самых общих ошибок программирования или только получить представление об огромных усилиях, потребовавшихся для разработки .NET, эта книга — обязательное чтение.”

— Франческо Балена (Francesco Balena),
The VB Migration Partner Team (www.vbmigration.com),

Архитектор программного обеспечения, автор
и Региональный директор Microsoft, Италия

“Инфраструктуры являются ценными, но известно, насколько трудно их создать: каждое решение, принимаемое при их создании, должно быть приспособлено к простоте их правильного использования и затруднять их неправильное использование. Эта книга последовательно проведет вас через рекомендации, которые устраняют многие из основных трудностей, по поводу которых мы так часто говорим «мне жаль, что я не знал это ранее. Мне жаль, что я не читал эту книгу ранее.»”

— Пол Бесли (Paul Besly), Главный технолог QA

“После книги Брукса *Мифический человек-месяц* самый главный разработчик программ своего времени написал книгу, столь полную важных советов для современного разработчика программ. У этой книги есть постоянное место на моей книжной полке, и я часто консультируюсь с ней.”

— Джордж Беркит, старший инженер-программист Genomic Solutions

“Обновленная с учетом новых языковых возможностей .NET Framework 3.0 и 3.5, эта книга продолжает быть критическим ресурсом для разработчиков и архитекторов .NET, которые проектируют инфраструктуры — библиотеки классов. Некоторые из существующих рекомендаций были расширены новыми замечаниями и большим количеством деталей, а также новыми советами, освещающими также такие особенности, как методы расширения и пустые (nullable) типы. Руководство поможет любому разработчику писать более чистый и более понятный код, в то время как замечания дают неоценимое понимание некоторых из решений разработчиков, которые сделали .NET Framework тем, чем она является сегодня.”

— Скотт Дорман, Microsoft MVP и Президент Международной ассоциации архитекторов программного обеспечения (Tampa Bay International Association of Software Architects)

“Содержащая информацию, полезную для разработчиков и архитекторов всех уровней, эта книга предоставляет практические рекомендации и самую важную информацию, которая помогает понять правила. Рекомендации по разработке инфраструктуры поднимают уже изданные рекомендации на более высокий уровень, а это необходимо, чтобы писать приложения, которые хорошо интегрируются в области .NET.”

— Кристоф Фальк, инженер-программист

“Чтение этой книги — насущная необходимость для всех .NET-разработчиков. Она дает ясные советы, как делать и чего не делать при проектировании библиотеки классов для .NET. Она также помогает понять дизайн и принципы создания .NET, что действительно помогает разработчикам понять причины, почему вещи являются такими, какими они есть. Эта книга поможет разработчикам проектировать их собственные библиотеки классов, а также поможет им более эффективно использовать библиотеку классов .NET в своих интересах.”

— Джеффри Рихтер, Автор/тренер/консультант, Wintellect

“Второе издание книги дает новое, важное понимание проектирования ваших собственных библиотек классов: Абрамс и Цвалина откровенно обсуждают добавление новых возможностей к новым версиям их продуктов с минимальным воздействием на написанный ранее код. Вы найдете замечательные примеры того, как создать версию N+1 вашего программного обеспечения, изучая, как группа, разработавшая библиотеки классов .NET, создала версии 2.0, 3.0 и 3.5 библиотеки .NET. Они смогли добавить универсальные средства, WCF, WPF, WF, а также LINQ с минимальным воздействием на существующие API, даже обеспечивая возможности для клиентов, желающих использовать только некоторые из новых возможностей, и поддерживая совместимость с первоначальной библиотекой.”

— Билл Вагнер (Wagner), Основатель и консультант SRT Solutions, автор книг Effective C# и More Effective C#

“Эта книга — необходимое чтение для всех архитекторов и разработчиков программ, думающих об инфраструктурах. Книга помогает понять некоторые решающие факторы дизайна .NET Framework. Нужно полагать, что это обязательное чтение для любого программиста, работа которого связана с созданием прикладных инфраструктур.”

— Питер Winkler, старший программист, Balance Technology Inc.

*Моей жене Эле за ее поддержку
в течение длинного процесса написания этой книги
и моим родителям Ядвиге и Янушу за их поддержку.*

Кржиштоф Цвалина



*Моей жене Тамаре.
Твоя любовь и терпение придают мне силу.*

Брэд Абрамс





Предисловие

При первом знакомстве с технологией .NET Framework я был очарован ею. Преимущества общезыковой исполняющей среды (Common Language Runtime, CLR), ее обширный API и язык C# были заметны с первого взгляда. Но в то же время был заметен и общий дизайн API, и ряд соглашений, которые использовались повсюду. Это было культурой .NET. Изучив лишь часть этой технологии, можно было без труда перенести это знание на другие области инфраструктуры Framework.

В течение 16 лет я работал над программным обеспечением с открытым исходным кодом. Так как разработчики не только охватили множественные платформы, но и многие годы в своей работе придерживались одного и того же стиля, соглашения о кодировании всегда были очень важны. Сотрудники службы сопровождения и поддержки обычно переписывают или адаптируют дополнения к программному обеспечению, чтобы гарантировать, что код соответствует стандартам кодирования и стилю проекта. Несомненно, лучше, если и разработчики программного проекта, и все, кто работает с ним, следуют соглашениям, используемым в уже существующем проекте. Чем подробнее будут описаны опыт и стандарты, тем проще будет будущим разработчикам разогнаться до нужной скорости реализации проекта. Это помогает в проекте соединить старый код с новым.

По мере роста .NET Framework, были идентифицированы новые действия, шаблоны и соглашения. Брэд и Кржиштоф записали всю эту новую информацию в настоящие рекомендации. Они создали блог, посвященный новым соглашениям, и, благодаря обратной связи с разработчиками, стали следовать их рекомендациям. По моему мнению, их блоги — документы, входящие в список обязательного чтения для всех, кто интересуется .NET Framework.

Первое издание этой книги стало настоящей классикой по двум важным причинам. Во-первых, оно позволило понять, почему и как реализованы различные .NET API. Во-вторых, его оценили за бесценные рекомендации, которым мы также стремились следовать в наших собственных программах и библиотеках. Новое издание основывается не только на успехе первого, но и содержит новые рекомендации, которые

18 ■ ■ Предисловие

были разработаны с тех пор. Аннотации к рекомендациям предоставлены ведущими архитекторами и программистами .NET, которые помогли сформулировать данные соглашения.

В заключение отмечу, что эта книга гораздо глубже, чем просто рекомендации. Эта классическая книга поможет стать лучшим программистом, а таких совсем немного в нашей отрасли промышленности.

Мигель де Икаса (Miguel de Icaza)
Бостон, Массачусетс



Предисловие к первому изданию

На начальном этапе разработки .NET Framework, еще до того как она получила свое название, я провел бесчисленные часы с членами групп разработки, делая обзор дизайнов, чтобы гарантировать, что конечным результатом будет последовательная платформа. Я всегда чувствовал, что ключевой характеристикой инфраструктуры должна быть непротиворечивость. Как только вы поняли одну часть инфраструктуры, другие части должны стать знакомыми немедленно.

Как и следовало ожидать, было много различных мнений, ведь ничто не может заечь живые и горячие дебаты так, как соглашения о кодировании. Однако, во имя непротиворечивости, мы постепенно отказывались от различий и шифровали результат в единый набор рекомендаций, которые позволяют программистам легко понимать и использовать инфраструктуру Framework.

Брэд Абрамс, а позже и Кржиштоф Цвалина помогли зафиксировать эти рекомендации в документе, который непрерывно обновлялся в течение шести лет. Книга, которую вы держите, является результатом их работы.

Рекомендации хорошо послужили нам на протяжении разработки трех версий .NET Framework и многочисленных меньших проектов, и они помогают вести разработку следующего поколения API для операционной системы Windows фирмы Microsoft.

Я надеюсь, что эта книга поможет вам достичь успеха в создании ваших собственных инфраструктур, библиотек классов, а также компонентов, простых для познания и легких в использовании.

Удачи и счастливых разработок проектов!
Андерс Хейльсберг (Anders Hejlsberg)
Редмонд, Вашингтон
Июнь, 2005

