# Предисловие

При первом знакомстве с технологией .NET Framework я был очарован ею. Преимущества общеязыковой 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 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 зажечь 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

# Предисловие

В этой книге собраны лучшие методы проектирования инфраструктур, которые являются объектно-ориентированными библиотеками многократного использования. Рекомендации применимы к различным инфраструктурам многократного использования.

- Большие системные инфраструктуры, такие как .NET Framework, которые обычно состоят из тысяч типов и используются миллионами разработчиков.
- Слои среднего размера для многократного использования при построении больших распределенных приложений или расширений системных инфраструктур, таких как расширения веб-служб (Web Services Enhancements).
- Маленькие компоненты, совместно используемые несколькими приложениями, такие как библиотека управления сеткой.

Стоит отметить, что эта книга сосредоточивается на проблемах разработки, которые непосредственно затрагивают программирование инфраструктуры (создание общедоступных API[1]). В результате мы вообще не рассматриваем многие подробности реализации. Поскольку данная книга посвящена разработке пользовательского интерфейса, в ней не рассматриваются подробности тестирования и не описывается, как реализовать, например, двоичную сортировку. Такой взгляд позволяет нам обсудить с проектировщиками инфраструктуры основные понятия, а не представить еще одну книгу по программированию.

Данные рекомендации были созданы на начальном этапе разработки .NET Framework. Сначала они представляли собой маленький набор соглашений об именовании и дизайне, но затем этот набор увеличивался, тщательно исследовался и был усовершенствован до такой степени, что в Microsoft их вообще считают каноническим способом проектирования инфраструктуры. Они воплощают опыт и совокупную мудрость разработчиков, потративших много часов на разработку более чем трех версий

---

[1] Это включает общедоступные типы и их общедоступные, защищенные и явно реализованные члены этих типов.

.NET Framework. Мы попытались избежать вывода рекомендаций из небольшого количества идеалистических основных положений по разработке программ, и мы думаем, что ежедневное использование текста группами разработчиков в Microsoft сделало данную книгу весьма прагматической.

Книга содержит много замечаний, которые объясняют компромиссы, объясняют хронологию, усиливают или критикуют рекомендации. Эти замечания написаны опытными проектировщиками инфраструктуры, экспертами промышленности, а также пользователями. Фактически это истории из опыта разработчиков, которые добавляют цвет ко многим представленным рекомендациям и объясняют их происхождение.

В тексте книги названия пространств имен, классы, интерфейсы, методы, свойства и типы выделены шрифтом фиксированной ширины.

Предполагается, что вы знакомы с основами программирования для .NET Framework, а также с особенностями, введенными в версии 3.5 инфраструктуры Framework. Если вы ищете хорошее введение в программирование инфраструктуры Framework, список литературы в конце книги содержит некоторые превосходные предложения.

## Представление рекомендаций

Рекомендации в книге представлены следующим образом: **НАСТОЯТЕЛЬНО РЕКОМЕНДУЕМ**, **РЕКОМЕНДУЕМ**, **НЕ РЕКОМЕНДУЕМ** и **НАСТОЯТЕЛЬНО НЕ РЕКОМЕНДУЕМ**. Каждое правило описывает хорошую или плохую практику, и у всех есть непротиворечивое представление. Перед хорошими (рекомендуемыми) действиями стоит ✓, а перед плохими (нерекомендуемыми) ✗. Формулировка каждой рекомендации также указывает, насколько сильна рекомендация. Например, рекомендация **НАСТОЯТЕЛЬНО РЕКОМЕНДУЕМ** должна выполняться всегда[2] (далее все примеры взяты из этой книги).

✓ **НАСТОЯТЕЛЬНО РЕКОМЕНДУЕМ** имена пользовательских классов-атрибутов заканчивать суффиксом “Attribute”.

```
public class ObsoleteAttribute : Attribute { ... }
```

Рекомендация **РЕКОМЕНДУЕМ** должна выполняться, если вы полностью понимаете ее, но если вы имеете серьезное основание не следовать ей по той или иной причине, не должны чувствовать себя плохо при нарушении этого правила.

✓ **РЕКОМЕНДУЕМ** определять структуры вместо классов, если экземпляры класса (типа) являются маленькими и обычно существуют недолго или внедряются в другие объекты.

---

[2] Рекомендации, которые должны выполняться всегда буквально, встречаются чрезвычайно редко. С другой стороны, у вас, вероятно, должен быть действительно необычный случай для того, чтобы отвергнуть рекомендацию НАСТОЯТЕЛЬНО РЕКОМЕНДУЕМ и получить решение, выгодное для пользователей инфраструктуры.

Рекомендация **НАСТОЯТЕЛЬНО НЕ РЕКОМЕНДУЕМ** указывает на то, что вы не должны делать почти никогда.

✗ **НАСТОЯТЕЛЬНО НЕ РЕКОМЕНДУЕМ** присваивать экземпляры изменяющихся типов полям только для чтения.

Рекомендация **НЕ РЕКОМЕНДУЕМ** указывает на нечто, вообще говоря, не очень хорошее, но есть известные случаи, в которых нарушение этого правила имеет смысл и может быть оправдано.

✗ **НЕ РЕКОМЕНДУЕМ** использовать `ICollection<T>` или `ICollection` как параметр только для того, чтобы обратиться к свойству `Count`.

Некоторые более сложные рекомендации сопровождаются дополнительной основной информацией, иллюстративными фрагментами кода и объяснением.

✓ **НАСТОЯТЕЛЬНО РЕКОМЕНДУЕМ** реализовывать `IEquatable<T>` на типах значений.

Метод `Object.Equals` на типах значений вызывает упаковку, и его заданная по умолчанию реализация не очень эффективна, потому что она использует рефлексию. Метод `IEquatable<T>.Equals` может предложить намного лучшую производительность и может быть реализован так, что не будет вызывать упаковку.

```
public struct Int32 : IEquatable<Int32> {
    public bool Equals(Int32 other){ ... }
}
```

## Выбор языка и примеров кода

Общеязыковая исполняющая среда (CLR) должна поддерживать множество языков программирования, включая как языки, реализации которых предоставлены Microsoft, такие как C++, VB, C#, F #, Python и Ruby, так и языки сторонних разработчиков, такие как язык Eiffel, КОБОЛ, ФОРТРАН и др. Поэтому книга была написана так, чтобы быть применимой к широкому набору языков, которые могут использоваться для разработки и применения современных инфраструктур.

Чтобы подкрепить месседж о многоязычном дизайне инфраструктуры, мы могли бы приводить примеры кода на нескольких языках программирования. Однако мы отказались от этого. Мы чувствовали, что использование различных языков поможет донести философский месседж, но это могло потребовать от читателей изучения нескольких новых языков, что не является целью этой книги.

Мы решили выбрать единственный язык, который, наиболее вероятно, будет читаемым самым широким диапазоном разработчиков. Мы выбрали C#, потому что это самый простой язык из семейства языков C (C, C++, Java и C#), семейства с богатой историей при разработке инфраструктуры.

Такой выбор языка близок сердцам многих разработчиков, и мы приносим извинения тем, кому наш выбор доставил неудобства.

## Об этой книге

Эта книга предлагает рекомендации по разработке инфраструктуры методом “сверху вниз”.

Глава 1, “Введение”, описывает общую философию разработки инфраструктуры. Это единственная глава без рекомендаций.

Глава 2, “Основные принципы разработки инфраструктуры”, предлагает принципы и рекомендации, которые фундаментальны для всей разработки инфраструктуры.

Глава 3, “Рекомендации по обозначениям”, содержит общие идиомы разработки и рекомендации по обозначению различных частей инфраструктуры, таких как пространства имен, типы и члены.

Глава 4, “Рекомендации по разработке типов”, содержит рекомендации по общей разработке типов.

Глава 5, “Проектирование членов”, детализирует предыдущий шаг и предоставляет рекомендации по разработке членов типов.

Глава 6, “Проектирование с целью расширяемости”, представляет проблемы и рекомендации, которые важны для того, чтобы гарантировать соответствующую расширяемость вашей инфраструктуры.

Глава 7, “Исключения”, представляет рекомендации по работе с исключениями и привилегированными механизмами сообщений об ошибках.

Глава 8, “Рекомендации по использованию”, содержит рекомендации по расширению и использованию типов, которые обычно содержатся в инфраструктурах.

Глава 9, “Общие шаблоны проектирования”, предлагает рекомендации и примеры дизайна общих шаблонов инфраструктуры.

Приложение А, “Соглашения о стиле кодирования в C#”, содержит краткое описание соглашений о кодировании, используемых в этой книге.

Приложение Б, “Использование FxCop для проверки рекомендаций по разработке инфраструктур”, описывает инструментальное средство FxCop. Этот инструмент помогает проанализировать двоичные инфраструктуры на согласованность с рекомендациями, описанными в этой книге. Этот инструмент включен в цифровой видеодиск, прилагаемый к этой книге.

Приложение В, “Пример спецификации API”, является примером спецификации API, которую создали проектировщики инфраструктуры в Microsoft, проектируя API.

Прилагаемый к книге цифровой видеодиск содержит несколько часов видеопрезентаций, относящихся к темам, представленным авторами в этой книге, пример типовой спецификации API, а также другие полезные ресурсы.