Введение

Авторы и читатели — одна команда

Авторам книг по технологиям приходится писать для очень требовательной группы людей (по вполне понятным причинам). Вам известно, что построение программных решений с применением любой платформы или языка исключительно сложно и специфично для отдела, компании, клиентской базы и поставленной задачи. Возможно, вы работаете в индустрии электронных публикаций, разрабатываете системы для правительства или местных органов власти либо сотрудничаете с NASA или какой-то военной отраслью. Вместе мы трудимся в нескольких отраслях, включая разработку обучающего ПО для детей (Oregon Trail/Amazon Trail), разнообразных производственных систем и проектов в медицинской и финансовой сферах. Написанный вами код на месте вашего трудоустройства почти на 100% будет иметь мало общего с кодом, который мы создавали на протяжении многих лет.

По указанной причине в книге мы намеренно решили избегать демонстрации примеров кода, свойственного какой-то конкретной отрасли или направлению программирования. Таким образом, мы объясняем язык C#, объектно-ориентированное программирование, .NET Runtime и библиотеки базовых классов .NET Core с использованием примеров, не привязанных к отрасли. Вместо того чтобы заставлять каждый пример наполнять сетку данными, подчитывать фонд заработной платы или выполнять другую задачу, специфичную для предметной области, мы придерживаемся темы, близкой каждому из нас: автомобили (с добавлением умеренного количества геометрических структур и систем расчета заработной платы для сотрудников). И вот тут наступает ваш черед.

Наша работа заключается в как можно лучшем объяснении языка программирования C# и основных аспектов платформы .NET 5. Мы также будем делать все возможное для того, чтобы снарядить вас инструментами и стратегиями, которые необходимы для продолжения обучения после завершения работы с данной книгой.

Ваша работа предусматривает усвоение этой информации и ее применение к решению своих задач программирования. Мы полностью отдаем себе отчет, что ваши проекты с высокой вероятностью не будут связаны с автомобилями и их дружественными именами, но именно в том и состоит суть прикладных знаний.

Мы уверены, что после освоения тем и концепций, представленных в настоящей книге, вы сможете успешно строить решения .NET 5, которые соответствуют вашей конкретной среде программирования.

Краткий обзор книги

Книга логически разделена на девять частей, каждая из которых содержит связанные друг с другом главы. Ниже приведено краткое содержание частей и глав.

Часть I. Язык программирования C# и платформа .NET 5

Эта часть книги предназначена для ознакомления с природой платформы .NET 5 и различными инструментами разработки, которые используются во время построения приложений .NET 5.

Глава 1. Введение в C# и .NET (Core) 5

Первая глава выступает в качестве основы для всего остального материала. Ее основная цель в том, чтобы представить вам набор строительных блоков .NET Core, таких как исполняющая среда .NET Runtime, общая система типов CTS, общеязыковая спецификация CLS и библиотеки базовых классов (BCL). Здесь вы впервые взглянете на язык программирования C#, пространства имен и формат сборок .NET 5.

Глава 2. Создание приложений на языке C#

Целью этой главы является введение в процесс компиляции файлов исходного кода C#. После установки .NET 5 SDK и исполняющей среды вы узнаете о совершенно бесплатном (но полнофункциональном) продукте Visual Studio Community, а также об исключительно популярном (и тоже бесплатном) продукте Visual Studio Code. Вы научитесь создавать, запускать и отлаживать приложения .NET 5 на языке C# с использованием Visual Studio и Visual Studio Code.

Часть II. Основы программирования на C#

Темы, представленные в этой части книги, очень важны, поскольку они связаны с разработкой ПО .NET 5 любого типа (например, веб-приложений, настольных приложений с графическим пользовательским интерфейсом, библиотек кода, служб и т.д.). Здесь вы узнаете о фундаментальных типах данных .NET 5, освоите манипулирование текстом и ознакомитесь с ролью модификаторов параметров C# (включая необязательные и именованные аргументы).

Глава 3. Главные конструкции программирования на C#: часть 1

В этой главе начинается формальное исследование языка программирования C#. Здесь вы узнаете о роли метода `Main()`, операторах верхнего уровня (нововведение в версии C# 9.0) , а также о многочисленных деталях, касающихся внутренних типов данных платформы .NET 5 и объявления переменных. Вы будете манипулировать текстовыми данными с применением типов `System.String` и `System.Text.StringBuilder`. Кроме того, вы исследуете итерационные конструкции и конструкции принятия решений, сопоставление с образцом, сужающие и расширяющие операции и ключевое слово `unchecked`.

Глава 4. Главные конструкции программирования на C#: часть 2

В этой главе завершается исследование ключевых аспектов C#, начиная с создания и манипулирования массивами данных. Затем вы узнаете, как конструировать перегруженные методы типов и определять параметры с применением ключевых слов `out`, `ref` и `params`. Также вы изучите типы перечислений, структуры и типы, допускающие `null`, плюс уясните отличие между типами значений и ссылочными типами. Наконец, вы освоите кортежи — средство, появившееся в C# 7 и обновленное в C# 8.

Часть III. Объектно-ориентированное программирование на C#

В этой части вы изучите основные конструкции языка C#, включая детали объектно-ориентированного программирования. Здесь вы научитесь обрабатывать исключения времени выполнения и взаимодействовать со строго типизированными интерфейсами. Вы также узнаете о времени существования объектов и сборке мусора.

Глава 5. Инкапсуляция

В этой главе начинается рассмотрение концепций объектно-ориентированного программирования (ООП) на языке C#. После представления главных принципов ООП (инкапсуляции, наследования и полиморфизма) будет показано, как строить надежные типы классов с использованием конструкторов, свойств, статических членов, констант и полей только для чтения. Вы также узнаете об определениях частичных типов, синтаксисе инициализации объектов и автоматических свойств, а в заключение главы будут рассматриваться типы записей, появившиеся в C# 9.0.

Глава 6. Наследование и полиморфизм

Здесь вы ознакомитесь с оставшимися главными принципами ООП (наследованием и полиморфизмом), которые позволяют создавать семейства связанных типов классов. Вы узнаете о роли виртуальных и абстрактных методов (и абстрактных базовых классов), а также о природе полиморфных интерфейсов. Затем вы исследуете сопоставление с образцом посредством ключевого слова `is` и в заключение выясните роль первичного базового класса платформы .NET Core — `System.Object`.

Глава 7. Структурированная обработка исключений

В этой главе обсуждаются способы обработки в кодовой базе аномалий, возникающих во время выполнения, за счет использования структурированной обработки исключений. Вы узнаете не только о ключевых словах C#, которые дают возможность решать такие задачи (`try`, `catch`, `throw`, `when` и `finally`), но и о разнице между исключениями уровня приложения и уровня системы. Вдобавок в главе будет показано, как настроить инструмент Visual Studio на прерывание для всех исключений, чтобы отлаживать исключения, оставшиеся без внимания.

Глава 8. Работа с интерфейсами

Материал этой главы опирается на ваше понимание объектно-ориентированной разработки и посвящен программированию на основе интерфейсов. Здесь вы узнаете, каким образом определять классы и структуры, поддерживающие несколько линий поведения, обнаруживать такие линии поведения во время выполнения и выборочно скрывать какие-то из них с применением явной реализации интерфейсов. В дополнение к созданию специальных интерфейсов вы научитесь реализовывать стандартные интерфейсы, доступные внутри платформы .NET Core, и использовать их для построения объектов, которые могут сортироваться, копироваться, перечисляться и сравниваться.

Глава 9. Время существования объектов

В финальной главе этой части исследуется управление памятью средой .NET Runtime с использованием сборщика мусора .NET Core. Вы узнаете о роли корневых элементов приложения, поколений объектов и типа `System.GC`. После пред-

ставления основ будут рассматриваться темы освобождаемых объектов (реализующих интерфейс `IDisposable`) и процесса финализации (с применением метода `System.Object.Finalize()`). В главе также описан класс `Lazy<T>`, позволяющий определять данные, которые не будут размещаться в памяти вплоть до поступления запроса со стороны вызывающего кода. Вы увидите, что такая возможность очень полезна, когда нежелательно загромождать кучу объектами, которые в действительности программе не нужны.

Часть IV. Дополнительные конструкции программирования на C#

В этой части книги вы углубите знания языка C# за счет исследования нескольких более сложных (и важных) концепций. Здесь вы завершите ознакомление с системой типов .NET Core, изучив коллекции и обобщения. Вы также освоите несколько более сложных средств C# (такие как методы расширения, перегрузка операций, анонимные типы и манипулирование указателями). Затем вы узнаете о делегатах и лямбда-выражениях, взглянете на язык LINQ, а в конце части ознакомитесь с процессами и многопоточным/асинхронным программированием.

Глава 10. Коллекции и обобщения

В этой главе исследуется тема *обобщений*. Вы увидите, что программирование с обобщениями предлагает способ создания типов и членов типов, которые содержат заполнители, указываемые вызывающим кодом. По существу обобщения значительно улучшают производительность приложений и безопасность в отношении типов. Здесь не только описаны разнообразные обобщенные типы из пространства имен `System.Collections.Generic`, но также показано, каким образом строить собственные обобщенные методы и типы (с ограничениями и без).

Глава 11. Расширенные средства языка C#

В этой главе вы сможете углубить понимание языка C# за счет исследования нескольких расширенных приемов программирования. Здесь вы узнаете, как перегружать операции и создавать специальные процедуры преобразования (явного и неявного) для типов. Вы также научитесь строить и взаимодействовать с индексаторами типов и работать с расширяющими методами, анонимными типами, частичными методами и указателями C#, используя контекст небезопасного кода.

Глава 12. Делегаты, события и лямбда-выражения

Целью этой главы является прояснение типа делегата. Выражаясь просто, делегат .NET Core представляет собой объект, который указывает на определенные методы в приложении. С помощью делегатов можно создавать системы, которые позволяют многочисленным объектам участвовать в двухстороннем взаимодействии. После исследования способов применения делегатов .NET Core вы ознакомитесь с ключевым словом `event` языка C#, которое упрощает манипулирование низкоуровневыми делегатами в коде. В завершение вы узнаете о роли лямбда-операции C# (`=>`), а также о связи между делегатами, анонимными методами и лямбда-выражениями.

Глава 13. LINQ to Objects

В этой главе начинается исследование языка интегрированных запросов (LINQ). Язык LINQ дает возможность строить строго типизированные выражения запро-

сов, которые могут применяться к многочисленным целевым объектам LINQ для манипулирования данными в самом широком смысле этого слова. Здесь вы изучите API-интерфейс LINQ to Objects, который позволяет применять выражения LINQ к контейнерам данных (например, массивам, коллекциям и специальным типам). Приведенная в главе информация будет полезна позже в книге при рассмотрении других API-интерфейсов.

Глава 14. Процессы, домены приложений и контексты загрузки

Опираясь на хорошее понимание вами сборок, в этой главе подробно раскрывается внутреннее устройство загруженной исполняемой сборки .NET Core. Целью главы является иллюстрация отношений между процессами, доменами приложений и контекстными границами. Упомянутые темы формируют основу для главы 15, где будет исследоваться конструирование многопоточных приложений.

Глава 15. Многопоточное, параллельное и асинхронное программирование

Эта глава посвящена построению многопоточных приложений. В ней демонстрируются приемы, которые можно использовать для написания кода, безопасного к потокам. Глава начинается с краткого напоминания о том, что собой представляет тип делегата .NET Core, и объяснения внутренней поддержки делегата для асинхронного вызова методов. Затем рассматриваются типы из пространства имен `System.Threading` и библиотека параллельных задач (Task Parallel Library — TPL). С применением TPL разработчики могут строить приложения .NET Core, которые распределяют рабочую нагрузку по всем доступным процессорам в исключительно простой манере. В главе также раскрыта роль API-интерфейса Parallel LINQ, который предлагает способ создания запросов LINQ, масштабируемых среди множества процессорных ядер. В завершение главы исследуется создание неблокирующих вызовов с использованием ключевых слов `async/await`, введенных в версии C# 5, локальных функций и обобщенных возвращаемых типов `async`, появившихся в версии C# 7, а также асинхронных потоков, добавленных в версии C# 8.

Часть V. Программирование с использованием сборок .NET Core

Эта часть книги посвящена деталям формата сборок .NET Core. Здесь вы узнаете не только о том, как развертывать и конфигурировать библиотеки кода .NET Core, но также о внутреннем устройстве двоичного образа .NET Core. Будет описана роль атрибутов .NET Core и распознавания информации о типе во время выполнения. Кроме того, объясняется роль исполняющей среды динамического языка (DLR) и ключевого слова `dynamic` языка C#. В последней главе части рассматривается синтаксис языка CIL и обсуждается роль динамических сборок.

Глава 16. Построение и конфигурирование библиотек классов

На самом высоком уровне термин “сборка” применяется для описания двоичного файла, созданного с помощью компилятора .NET Core. Однако в действительности понятие сборки намного шире. Вы научитесь создавать и развертывать сборки и узнаете, в чем отличие между библиотеками классов и консольными приложениями, а также между библиотеками классов .NET Core и .NET Standard. В конце главы раскрываются новые возможности, доступные в .NET 5, такие как однофайловое автономное развертывание.

Глава 17. Рефлексия типов, позднее связывание и программирование на основе атрибутов

В этой главе продолжается исследование сборок .NET Core. Здесь будет показано, как обнаруживать типы во время выполнения с использованием пространства имен `System.Reflection`. Посредством типов из упомянутого пространства имен можно строить приложения, способные считывать метаданные сборки на лету. Вы также узнаете, как загружать и создавать типы динамически во время выполнения с применением позднего связывания. Напоследок в главе обсуждается роль атрибутов .NET Core (стандартных и специальных). Для закрепления материала в главе демонстрируется построение расширяемого приложения с подключаемыми оснастками.

Глава 18. Динамические типы и среда DLR

В версии .NET 4.0 появился новый аспект исполняющей среды .NET, который называется *исполняющей средой динамического языка* (DLR). Используя DLR и ключевое слово `dynamic` языка C#, можно определять данные, которые в действительности не будут распознаваться вплоть до времени выполнения. Такие средства существенно упрощают решение ряда сложных задач программирования для .NET Core. В этой главе вы ознакомитесь со сценариями применения динамических данных, включая использование API-интерфейсов рефлексии .NET Core и взаимодействие с унаследованными библиотеками COM с минимальными усилиями.

Глава 19. Язык CIL и роль динамических сборок

В последней главе этой части преследуется двойная цель. В первой половине главы рассматривается синтаксис и семантика языка CIL, а во второй — роль пространства имен `System.Reflection.Emit`. Типы из указанного пространства имен можно применять для построения ПО, которое способно генерировать сборки .NET Core в памяти во время выполнения. Формально сборки, которые определяются и выполняются в памяти, называются *динамическими сборками*.

Часть VI. Работа с файлами, сериализация объектов и доступ к данным

К настоящему моменту вы уже должны хорошо ориентироваться в языке C# и в подробностях формата сборок .NET Core. В данной части книги ваши знания расширяются исследованием нескольких часто используемых служб, которые можно обнаружить внутри библиотек базовых классов, включая файловый ввод-вывод, сериализация объектов и доступ к базам данных посредством ADO.NET.

Глава 20. Файловый ввод-вывод и сериализация объектов

Пространство имен `System.IO` позволяет взаимодействовать со структурой файлов и каталогов машины. В этой главе вы узнаете, как программно создавать (и удалять) систему каталогов. Вы также научитесь перемещать данные между различными потоками (например, файловыми, строковыми и находящимися в памяти). Кроме того, в главе рассматриваются службы сериализации объектов в формат XML и JSON платформы .NET Core. Сериализация позволяет сохранять состояние объекта (или набора связанных объектов) в потоке для последующего использования. Десериализация представляет собой процесс извлечения объекта из потока в память с целью потребления внутри приложения.

Глава 21. Доступ к данным с помощью ADO.NET

Эта глава посвящена доступу к данным с использованием ADO.NET — API-интерфейса доступа к базам данных для приложений .NET Core. В частности, здесь рассматривается роль поставщиков данных .NET Core и взаимодействие с реляционной базой данных с применением инфраструктуры ADO.NET, которая представлена объектами подключений, объектами команд, объектами транзакций и объектами чтения данных. Кроме того, в главе начинается создание уровня доступа к данным `AutoLot`, который будет расширен в главах 22 и 23.

Часть VII. Entity Framework Core

У вас уже есть четкое представление о языке C# и деталях формата сборок .NET Core. В этой части вы узнаете о распространенных службах, реализованных внутри библиотек базовых классов, в числе которых файловый ввод-вывод, доступ к базам данных с использованием ADO.NET и доступ к базам данных с применением Entity Framework Core.

Глава 22. Введение в Entity Framework Core

В этой главе рассматривается инфраструктура Entity Framework (EF) Core, которая представляет собой систему объектно-реляционного отображения (ORM), построенную поверх ADO.NET. Инфраструктура EF Core предлагает способ написания кода доступа к данным с использованием строго типизированных классов, напрямую отображаемых на бизнес-модель. Здесь вы освоите строительные блоки EF Core, включая `DbContext`, сущности, специализированный класс коллекции `DbSet<T>` и `DbChangeTracker`. Затем вы узнаете о выполнении запросов, отслеживаемых и неотслеживаемых сущностях, а также о других примечательных возможностях EF Core. В заключение рассматривается глобальный инструмент EF Core для интерфейса командной строки .NET Core (CLI).

Глава 23. Построение уровня доступа к данным с помощью Entity Framework Core

В этой главе создается уровень доступа к данным `AutoLot`. Глава начинается с построения шаблонов сущностей и производного от `DbContext` класса для базы данных `AutoLot` из главы 21. Затем подход “сначала база данных” меняется на подход “сначала код”. Сущности обновляются до своей финальной версии, после чего создается и выполняется миграция, чтобы обеспечить соответствие сущностям. Последнее изменение базы данных заключается в создании миграции для хранимой процедуры из главы 21 и нового представления базы данных. В целях инкапсуляции кода добавляются хранилища данных, и затем организуется процесс инициализации данных. Наконец, проводится испытание уровня доступа к данным с использованием инфраструктуры xUnit для автоматизированного интеграционного тестирования.

Часть VIII. Разработка клиентских приложений для Windows

Первоначальный API-интерфейс для построения графических пользовательских интерфейсов настольных приложений, поддерживаемый платформой .NET, назывался Windows Forms. Хотя он по-прежнему доступен, в версии .NET 3.0 программистам был предложен API-интерфейс под названием Windows Presentation Foundation (WPF). В отличие от Windows Forms эта инфраструктура для построения пользовательских

интерфейсов объединяет в единую унифицированную модель несколько основных служб, включая привязку данных, двумерную и трехмерную графику, анимацию и форматированные документы. Все это достигается с использованием декларативной грамматики разметки, которая называется расширяемым языком разметки приложений (XAML). Более того, архитектура элементов управления WPF предлагает легкий способ радикального изменения внешнего вида и поведения типового элемента управления с применением всего лишь правильно оформленной разметки XAML.

Глава 24. Введение в Windows Presentation Foundation и XAML

Эта глава начинается с исследования мотивации создания WPF (с учетом того, что в .NET уже существовала инфраструктура для разработки графических пользовательских интерфейсов настольных приложений). Затем вы узнаете о синтаксисе XAML и ознакомитесь с поддержкой построения приложений WPF в Visual Studio.

Глава 25. Элементы управления, компоновки, события и привязка данных в WPF

В этой главе будет показано, как работать с элементами управления и диспетчерами компоновки, предлагаемыми WPF. Вы узнаете, каким образом создавать системы меню, окна с разделителями, панели инструментов и строки состояния. Также в главе рассматриваются API-интерфейсы (и связанные с ними элементы управления), входящие в состав WPF, в том числе Ink API, команды, маршрутизируемые события, модель привязки данных и свойства зависимости.

Глава 26. Службы визуализации графики WPF

Инфраструктура WPF является API-интерфейсом, интенсивно использующим графику, и с учетом этого WPF предоставляет три подхода к визуализации графических данных: фигуры, рисунки и геометрические объекты, а также визуальные объекты. В настоящей главе вы ознакомитесь с каждым подходом и попутно изучите несколько важных графических примитивов (например, кисти, перья и трансформации). Кроме того, вы узнаете, как встраивать векторные изображения в графику WPF и выполнять операции проверки попадания в отношении графических данных.

Глава 27. Ресурсы, анимация, стили и шаблоны WPF

В этой главе освещены важные (и взаимосвязанные) темы, которые позволят углубить знания API-интерфейса WPF. Первым делом вы изучите роль логических ресурсов. Система логических ресурсов (также называемых объектными ресурсами) предлагает способ именования и ссылки на часто используемые объекты внутри приложения WPF. Затем вы узнаете, каким образом определять, выполнять и управлять анимационной последовательностью. Вы увидите, что применение анимации WPF не ограничивается видеоиграми или мультимедиа-приложениями. В завершение главы вы ознакомитесь с ролью стилей WPF. Подобно веб-странице, использующей CSS или механизм тем ASP.NET, приложение WPF может определять общий вид и поведение для целого набора элементов управления.

Глава 28. Уведомления WPF, проверка достоверности, команды и MVVM

Эта глава начинается с исследования трех основных возможностей инфраструктуры WPF: уведомлений, проверки достоверности и команд. В разделе, в котором рассматриваются уведомления, вы узнаете о наблюдаемых моделях и коллекциях, а

также о том, как они поддерживают данные приложения и пользовательский интерфейс в синхронизированном состоянии. Затем вы научитесь создавать специальные команды для инкапсуляции кода. В разделе, посвященном проверке достоверности, вы ознакомитесь с несколькими механизмами проверки достоверности, которые доступны для применения в приложениях WPF. Глава завершается исследованием паттерна "модель-представление-модель представления" (MVVM) и созданием приложения, демонстрирующего паттерн MVVM в действии.

Часть IX. ASP.NET Core

Эта часть посвящена построению веб-приложений с применением инфраструктуры ASP.NET Core, которую можно использовать для создания веб-приложений и служб REST.

Глава 29. Введение в ASP.NET Core

В этой главе обсуждается инфраструктура ASP.NET Core и паттерн MVC. Сначала объясняются функциональные средства, перенесенные в ASP.NET Core из классических инфраструктур ASP.NET MVC и Web API, в том числе контроллеры и действия, привязка моделей, маршрутизация и фильтры. Затем рассматриваются новые функциональные средства, появившиеся в ASP.NET Core, включая внедрение зависимостей, готовность к взаимодействию с облачными технологиями, осведомленная о среде система конфигурирования, шаблоны развертывания и конвейер обработки запросов HTTP. Наконец, в главе создаются два проекта ASP.NET Core, которые будут закончены в последующих двух главах, демонстрируются варианты запуска приложений ASP.NET Core и начинается процесс конфигурирования этих двух проектов ASP.NET Core.

Глава 30. Создание служб REST с помощью ASP.NET Core

В этой главе завершается создание приложения REST-службы ASP.NET Core. Первым делом демонстрируются разные механизмы возвращения клиенту результатов JSON и встроенная поддержка приложений служб, обеспечиваемая атрибутом `ApiController`. Затем добавляется пакет Swagger/OpenAPI, чтобы предоставить платформу для тестирования и документирования службы. В конце главы создаются контроллеры для приложения и фильтр исключений.

Глава 31. Создание приложений MVC с помощью ASP.NET Core

В последней главе книги заканчивается рассмотрение ASP.NET Core и работа над веб-приложением MVC. Сначала подробно обсуждаются представления и механизм представлений Razor, включая компоновки и частичные представления. Затем исследуются вспомогательные функции дескрипторов, а также управление библиотеками клиентской стороны и пакетирование/минификация этих библиотек. Далее завершается построение класса `CarsController` и его представлений вместе со вспомогательными функциями дескрипторов. В управляемое данными меню добавляется компонент представления и рассматривается шаблон параметров. Наконец, создается оболочка для службы клиента HTTP, а класс `CarsController` обновляется с целью использования службы ASP.NET Core вместо уровня доступа к данным `AutoLot`.

Ждем ваших отзывов!

Вы, читатель этой книги, и есть главный ее критик. Мы ценим ваше мнение и хотим знать, что было сделано нами правильно, что можно было сделать лучше и что еще вы хотели бы увидеть изданным нами. Нам интересны любые ваши замечания в наш адрес.

Мы ждем ваших комментариев и надеемся на них. Вы можете прислать нам электронное письмо либо просто посетить наш веб-сайт и оставить свои замечания там. Одним словом, любым удобным для вас способом дайте нам знать, нравится ли вам эта книга, а также выскажите свое мнение о том, как сделать наши книги более интересными для вас.

Отправляя письмо или сообщение, не забудьте указать название книги и ее авторов, а также свой обратный адрес. Мы внимательно ознакомимся с вашим мнением и обязательно учтем его при отборе и подготовке к изданию новых книг.

Наши электронные адреса:

E-mail: `info.dialektika@gmail.com`

WWW: `http://www.dialektika.com`