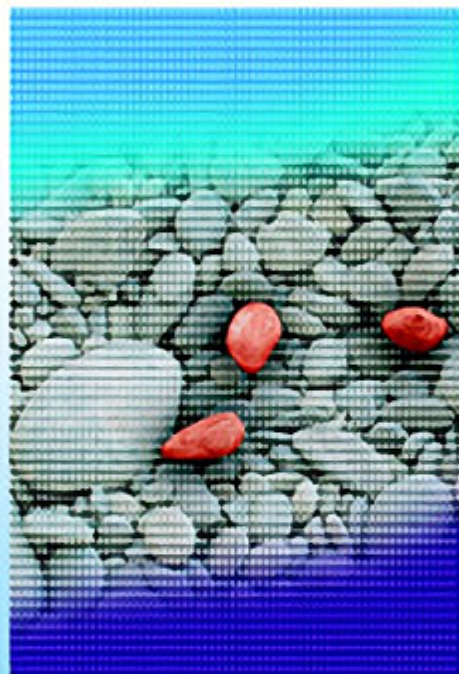


Алексей Голощапов



Microsoft®

Visual Studio 2010



- Разработка приложений с использованием технологий Windows Presentation Foundation, Windows Forms
- Создание веб-приложений с помощью ASP.NET, MVC, AJAX, jQuery, Silverlight
- Работа с базами данных с использованием технологий LINQ, Entity Framework, ASP.NET Dynamic Data
- Технология Windows Communication Foundation
- Управление рабочими процессами с помощью Windows Workflow Foundation
- Локализация и развертывание приложений

+  cd

Наиболее
полное
руководство

В ПОДЛИННИКЕ®

УДК 681.3.068
ББК 32.973.26-018.1
Г60

Голощанов А. Л.

Г60 Microsoft® Visual Studio 2010. — СПб.: БХВ-Петербург, 2011. — 544 с.: ил.
+ CD-ROM — (В подлиннике)

ISBN 978-5-9775-0617-5

Рассмотрены приемы работы в интегрированной среде разработки Microsoft Visual Studio 2010, а также новые технологии и элементы среды, предназначенные для создания современных приложений. Описана работа с решениями, проектами, редакторами и визуальными конструкторами. Описывается создание различных типов приложений: с помощью технологий Windows Presentation Foundation и Windows Forms, создание веб-приложений с помощью технологий ASP.NET, MVC, AJAX, jQuery, Silverlight. Рассматривается проектирование и развертывание баз данных, а также создание приложений для работы с базами данных с использованием технологий LINQ, Entity Framework, ASP.NET Dynamic Data, технология создания служб Windows Communication Foundation, управление рабочими процессами с помощью Windows Workflow Foundation, локализация и развертывание приложений. Материал книги сопровождается практическими примерами в тексте и на компакт-диске.

Для программистов

УДК 681.3.068
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Игорь Шишигин</i>
Зав. редакцией	<i>Григорий Добин</i>
Редактор	<i>Екатерина Капалыгина</i>
Компьютерная верстка	<i>Ольги Сергиенко</i>
Корректор	<i>Зинаида Дмитриева</i>
Дизайн серии	<i>Инны Тачиной</i>
Оформление обложки	<i>Елены Беляевой</i>
Зав. производством	<i>Николай Тверских</i>

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 31.01.11.

Формат 70×100^{1/16}. Печать офсетная. Усл. печ. л. 43,86.

Тираж 1500 экз. Заказ №

"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.60.953.Д.005770.05.09 от 26.05.2009 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0617-5

© Голощанов А. Л., 2011
© Оформление, издательство "БХВ-Петербург", 2011

Оглавление

Введение.....	1
На кого рассчитана эта книга.....	1
Краткое описание глав.....	2
Исходные коды примеров	5
Благодарности	5
ЧАСТЬ I. ОСНОВЫ VISUAL STUDIO 2010.....	7
Глава 1. Общие сведения о Visual Studio 2010.....	9
Версии Visual Studio 2010	9
Усовершенствования интегрированной среды разработки Visual Studio	10
Новая стартовая страница	10
Поддержка нескольких мониторов.....	11
Выделение ссылок в коде	11
Масштабирование	11
Выделение области	11
Тестирование и отладка приложений.....	14
Усовершенствования разработки приложений ASP.NET	14
Усовершенствования конструкторов для Windows Presentation Foundation и Silverlight	14
Новая версия Windows Workflow Foundation	16
Резюме	17
Глава 2. Настройка рабочей среды Visual Studio 2010	18
Параметры настройки среды	18
Миграция параметров настроек.....	20
<i>Start Page</i>	23
Документация по продукту	25
Резюме	26
Глава 3. Интегрированная среда разработки	27
Создание проектов.....	27
Шаблоны Visual Studio	28
Создание проекта	28

Решения	30
Добавление проектов в решение.....	30
Установка стартового проекта.....	31
Конфигурации <i>Debug</i> и <i>Release</i>	32
Выбор конфигурации.....	32
Редактирование конфигурации	33
Рефакторинг кода.....	34
Система окон.....	35
Расположение окон	36
Окно <i>Properties</i>	37
Окно <i>Class View</i>	38
Окно <i>Object Browser</i>	39
Окно <i>Code Definition</i>	40
Окно <i>Call Hierarchy</i>	41
Окно <i>Server Explorer</i>	41
Окно <i>Task List</i>	42
Редакторы кода	43
Форматирование кода.....	43
Отображение нумерации строк.....	47
Окно <i>Bookmarks</i>	48
Визуальные конструкторы и панели инструментов.....	49
Резюме	50
Глава 4. Отладка приложений	51
Основные типы ошибок	51
Синтаксические ошибки.....	51
Логические ошибки	52
Ошибки периода выполнения	52
Отладчик Visual Studio	54
Точки прерывания.....	55
Настройка точки прерывания.....	56
Окно <i>Breakpoints</i>	57
Настройка точки прерывания функции.....	58
Прерывание на основе условий	59
Окно <i>File Breakpoint</i>	59
Окно <i>Breakpoint Condition</i>	60
Окно <i>Breakpoint Hit Count</i>	61
Окно <i>Breakpoint Filter</i>	62
Окно <i>When Breakpoint Is Hit</i>	63
Пошаговое прохождение кода	64
Начало отладки приложения.....	64
Прохождение по коду.....	66
Продолжение отладки.....	66
Окончание отладки	67
Отладочные окна.....	67
Окно <i>Output</i>	67
Окно <i>Locals</i>	68
Окно <i>Autos</i>	69

Окно <i>Watch</i>	70
Окно <i>QuickWatch</i>	71
Окно <i>Command</i> в режиме <i>Immediate</i>	72
Всплывающие подсказки данных <i>DataTips</i>	72
Окна визуализации данных	73
Исключения	73
Классы <i>Debug</i> и <i>Trace</i>	75
Вывод трассировки в окно <i>Output</i>	75
Запись данных в набор <i>Listeners</i>	77
Трассировочные переключатели	80
Резюме	85
Глава 5. Создание приложений Windows Forms.....	86
Создание проекта Windows Forms Application	86
Иерархия классов Windows Forms	91
Свойства формы	93
Методы формы	94
Жизненный цикл и события формы	95
Создание обработчика событий	97
Компоновка и позиционирование элементов управления	97
Привязка и закрепление элементов	98
Приложения с несколькими формами	100
Назначение стартовой формы	100
Переключение между формами	101
Принципы создания пользовательского интерфейса	104
Меню	104
Панель инструментов	105
Строка состояния	108
Использование контейнерных элементов	108
Резюме	110
ЧАСТЬ II. ТЕХНОЛОГИИ ДОСТУПА К ДАННЫМ	111
Глава 6. Проектирование баз данных в Visual Studio 2010.....	113
Создание базы данных в Visual Studio	113
Определение таблиц	114
Создание диаграммы базы данных	117
Создание связей между таблицами	118
Разработка хранимых процедур	120
Отладка хранимых процедур	125
Шаблоны проектов баз данных	126
Автоматическое генерирование скриптов	132
Выполнение скриптов	133
Представление схемы	133
Создание тестовых данных	133
Настройка генераторов данных	135
Изменение свойств генератора	136
Создание данных	137
Резюме	138

Глава 7. Технология доступа к данным ADO.NET	139
Архитектура данных ADO.NET.....	139
Провайдеры данных.....	140
Организация доступа к данным	141
Объект <i>DataSet</i>	142
Подключение к базе данных	142
Объект <i>Connection</i>	144
Объект <i>Command</i>	145
Объект <i>DataReader</i>	146
Приложение для чтения данных	148
Передача параметров в объект <i>Command</i>	152
Использование типизированного объекта <i>DataReader</i>	153
Модификация данных.....	154
Резюме	163
Глава 8. Работа с автономными данными в ADO.NET.....	164
Объект <i>DataAdapter</i>	164
Взаимодействие объектов <i>DataAdapter</i> и <i>DataSet</i>	167
Реализация отображения при выборке данных	171
Объект <i>DataSet</i> со строгим контролем типов.....	174
Создание источника данных	175
Модификация данных в <i>DataSet</i>	180
Обновление базы данных	181
Реализация отдельного уровня данных.....	181
Динамическое связывание данных в период выполнения.....	183
Сортировка и фильтрация данных.....	190
Создание объекта <i>DataView</i>	190
Сортировка данных.....	190
Фильтрация данных	194
Объект <i>DataSet</i> и XML	198
Резюме	202
Глава 9. LINQ	203
Источники данных LINQ.....	203
LINQ to Objects	204
Основные операции запросов LINQ.....	204
Запросы к коллекциям	207
Запросы к пользовательским классам	209
LINQ to XML.....	211
<i>XElement</i>	212
Выполнение запросов LINQ to XML.....	214
LINQ to DataSet	216
Запросы к наборам данных с помощью LINQ to DataSet.....	217
Создание запросов LINQ to DataSet в приложениях.....	217
LINQ to SQL	221
Создание объектной модели LINQ to SQL	221
Создание приложения для работы с данными	225
Резюме	229

Глава 10. Entity Framework.....	230
Работа с данными в Entity Framework	230
Entity Data Model.....	231
Создание Entity Data Model в Visual Studio	231
Создание приложения для работы с данными.....	237
Комплексные типы	239
Импорт хранимых процедур из базы данных.....	239
Обновление модели хранения данных	239
Вызов хранимых процедур	241
Резюме	243
ЧАСТЬ III. СОЗДАНИЕ УРОВНЯ ПРЕЗЕНТАЦИЙ.....	245
Глава 11. Веб-формы ASP.NET	247
Жизненный цикл веб-страниц ASP.NET	247
Основные события веб-страницы	247
Обратная отсылка на сервер.....	249
Состояние вида.....	249
Веб-приложения и веб-сайты.....	250
Выбор места расположения веб-сайта	253
Веб-формы.....	256
Серверные элементы управления	258
Элементы управления HTML	259
Элементы управления Web	266
Базовые элементы управления Web	268
Элементы валидации данных.....	271
<i>RequiredFieldValidator</i>	273
<i>RangeValidator</i>	273
<i>CompareValidator</i>	274
<i>RegularExpressionValidator</i>	274
<i>CustomValidator</i>	275
Пример веб-страницы с валидацией введенных данных	275
<i>ValidationSummary</i>	277
Привязка данных к элементам управления Web	280
Использование элемента управления <i>GridView</i>	280
Обновление данных в <i>GridView</i>	283
Привязка данных к спискам	284
Динамическая привязка данных	286
Резюме	289
Глава 12. Стили и темы.....	290
Стили.....	290
Стили элементов	290
Стили страниц	291
Каскадные таблицы стилей	291
Создание и управление стилями	292
Темы.....	298
Создание темы.....	299

Подключение темы	299
Определение темы в файле конфигурации	301
Резюме	301
Глава 13. Мастер-страницы и управление навигацией	302
Мастер-страницы	302
Создание мастер-страницы	302
Создание страницы содержимого	307
Навигация	310
Элемент управления <i>TreeView</i>	310
Объекты <i>TreeNode</i>	311
Применение стилей к типам узлов	311
Элемент управления <i>Menu</i>	314
Стили элемента управления <i>Menu</i>	314
Резюме	317
Глава 14. ASP.NET AJAX	318
Архитектура AJAX	318
Элементы управления AJAX в ASP.NET	319
Создание страницы AJAX	320
Библиотека AJAX Control Toolkit	323
Подключение набора элементов AJAX Control Toolkit к панели <i>Toolbox</i>	324
Применение элементов управления AJAX	325
Расширения для элементов управления	327
Резюме	330
Глава 15. Библиотека jQuery	331
Подключение библиотеки jQuery	331
Объекты библиотеки jQuery	334
Селекторы	335
Использование селекторов и фильтров	336
Фильтры и эффекты	341
Операции над множествами	345
Создание элемента <i>Accordion</i>	346
Резюме	348
Глава 16. ASP.NET Dynamic Data	350
Архитектура платформы Dynamic Data	350
Создание веб-сайта с использованием ASP.NET Dynamic Data	351
Шаблоны страниц	353
Шаблоны сущностей	353
Шаблоны полей	354
Шаблоны фильтров	354
Элементы уровня данных	354
Регистрация модели данных	355
Резюме	359
Глава 17. ASP.NET MVC	360
Архитектура MVC	360

Создание приложения MVC.....	361
Выполнение запросов в MVC.....	364
Маршрутизация URL-адресов.....	366
Контроллеры и методы действий.....	367
Методы действий.....	369
Возвращаемый тип <i>ActionResult</i>	369
Параметры методов действий.....	370
Добавление нового контроллера.....	371
Добавление представления.....	373
Создание и привязка моделей.....	377
Резюме.....	381
Глава 18. Windows Presentation Foundation.....	382
Архитектура WPF.....	382
Типы приложений WPF.....	384
Создание приложения WPF.....	384
Компоновка окна приложения WPF.....	391
Контейнер <i>Grid</i>	392
Контейнер <i>UniformGrid</i>	394
Контейнер <i>Canvas</i>	395
Контейнер <i>DockPanel</i>	396
Контейнер <i>StackPanel</i>	397
Контейнер <i>WrapPanel</i>	398
Переключение между окнами.....	399
Работа WPF с документами.....	401
Создание проектов XВАР.....	404
Привязка данных.....	406
2D-графика.....	408
Двумерные формы.....	409
Шаблоны элементов управления.....	411
3D-графика.....	413
Окно просмотра.....	413
Материал поверхности.....	413
Источники света.....	414
Камера.....	415
Построение геометрической модели.....	415
Вращения.....	419
Резюме.....	424
Глава 19. Silverlight.....	425
Настройка среды разработки.....	425
Создание приложения Silverlight.....	426
Использование кода приложений WPF в Silverlight.....	429
Анимация в Silverlight.....	431
Трехмерные эффекты с трансформацией перспективы.....	434
Интеграция Silverlight с веб-страницей.....	437
Использование HTML.....	437
Использование Silverlight.js.....	438
Резюме.....	440

ЧАСТЬ IV. СЕРВИСЫ И КОММУНИКАЦИИ	441
Глава 20. Windows Communication Foundation	443
Создание проекта сервиса WCF	443
Создание сервиса для работы с данными	450
Тестирование сервиса WCF	460
Резюме	463
Глава 21. Windows Workflow Foundation	464
Компоненты Windows Workflow Foundation	464
Среда выполнения WWF	465
Взаимодействие между компонентами рабочего процесса	466
Архитектура действий	466
Жизненный цикл действия	467
Рабочие процессы и действия	467
Модель данных действия	468
Выгрузка и сохранение рабочих процессов	468
Создание проектов Windows Workflow Foundation	469
Визуальный конструктор WWF	471
Аргументы и переменные	471
Добавление действий	473
Отображение вывода рабочего процесса	477
Создание циклических процессов	478
Действие <i>While</i>	478
Действие <i>DoWhile</i>	481
Моделирование рабочих процессов с помощью блок-схем	482
Резюме	485
ЧАСТЬ V. ЛОКАЛИЗАЦИЯ И РАЗВЕРТЫВАНИЕ ПРИЛОЖЕНИЙ	487
Глава 22. Локализация приложений	489
Концепция культур	489
Локализация приложений Windows Forms	490
Создание локализованного приложения	491
Локализация веб-приложений	496
Резюме	501
Глава 23. Развертывание приложений	502
Windows Installer	502
Создание проекта развертывания	503
Параметры компоновки проекта	508
Регистрация компонентов приложения	510
Редакторы свойств установки	511
<i>File System Editor</i>	511
<i>Registry Editor</i>	513
<i>File Types Editor</i>	514
<i>User Interface Editor</i>	515

<i>Custom Actions Editor</i>	516
<i>Launch Conditions Editor</i>	517
Сборка пакета установки.....	518
Установка приложения.....	518
Резюме	519
Приложение. Описание компакт-диска и установка примеров.....	521
Предметный указатель	523

ГЛАВА 3



Интегрированная среда разработки

Чтобы эффективно управлять компонентами, используемыми на этапе разработки, например ссылками, подключениями данных, папками и файлами, в Visual Studio предусмотрены два типа контейнеров. Эти контейнеры называются *решениями* и *проектами*. Также Visual Studio предоставляет папки решений для того, чтобы структурировать связанные проекты по группам и затем выполнять действия над этими группами проектов. Частью интегрированной среды разработки является интерфейс для просмотра и управления этими контейнерами и связанными с ними элементами — **Solution Explorer**.

Создание проектов

Проект — это основная единица, с которой работает программист. Он выбирает тип проекта, а Visual Studio создает шаблон проекта в соответствии с выбранным типом.

В Visual Studio редко создается пустой проект, к которому затем добавляется код. Вместо этого вы указываете среде Visual Studio тип проекта, который вы хотите создать, и среда разработки генерирует файлы и программный код, которые служат основой для выбранного типа проекта. После этого вы можете работать над приложением, добавляя ваш код к созданному шаблону проекта. Например, если требуется создать проект, основанный на Windows Forms, то среда Visual Studio 2010 сгенерирует проект с пустой формой. Если потребуется создать консольное приложение, то Visual Studio сгенерирует файл `program.cs`, в котором уже будут основные пространства имен, класс `Program` и точка входа в приложение — статический метод `Main()`.

При создании проекта Visual Studio также определяется, должен ли проект компилироваться в консольное приложение, библиотеку классов, приложения Windows Forms, Windows Presentation Foundation и др. Выбранный тип проекта также указывает компилятору, какие внешние библиотеки необходимо подключить. Конечно, вы можете изменить все эти параметры настройки в процессе создания приложения.

Шаблоны Visual Studio

Ряд предопределенных шаблонов проекта и шаблонов элементов проекта устанавливаются при установке среды разработки Visual Studio. Эти шаблоны можно использовать для создания основного контейнера проекта и предварительного набора элементов, необходимых для разработки приложения, класса, элемента управления или библиотеки. Можно также использовать один из многих шаблонов элементов проекта для создания, например, приложения Windows Forms или страницы Web Forms, чтобы индивидуально изменять приложение по мере его разработки.

Шаблоны проектов Visual Studio предоставляют многократно используемую и настраиваемую основу для проектов и элементов, позволяющую ускорить процесс разработки приложений, поскольку избавляют программистов от необходимости создания новых проектов и элементов "с нуля".

Эти шаблоны предоставляют пользователям отправную точку для создания новых или расширения текущих проектов. Шаблоны проектов предоставляют файлы, необходимые для конкретного типа проекта, включают стандартные ссылки на сборки и задают свойства проекта и параметры компилятора по умолчанию. Шаблоны элементов могут варьироваться по сложности от одного пустого файла с правильным расширением имени до элементов из нескольких файлов, включая, например, файлы исходного кода с кодом-основой, файлы сведений о конструкторе и внедренные ресурсы.

В дополнение к установленным шаблонам можно разработать собственные шаблоны проектов или использовать шаблоны, созданные сторонними производителями. Все шаблоны проектов и элементов, независимо от того, поставляются они с Visual Studio или созданы сторонними производителями, работают одинаковым образом и состоят из следующих элементов:

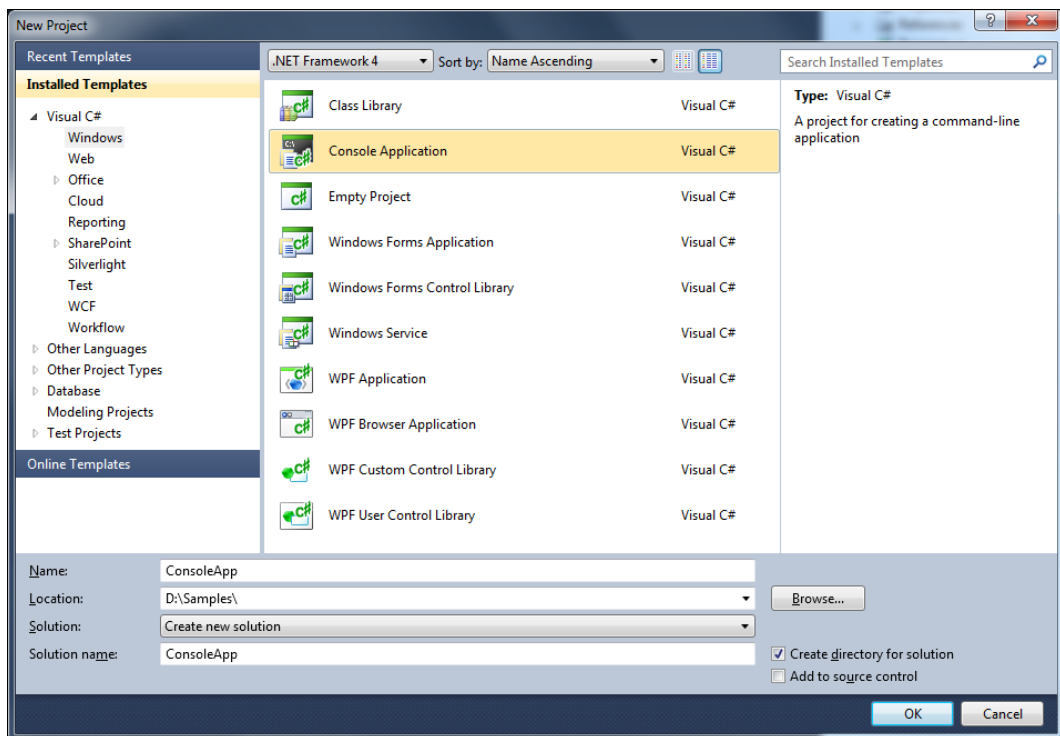
- ♦ файлов, которые будут созданы при выборе этого типа шаблона. Сюда входят все файлы с исходным кодом, внедренные ресурсы, файлы проекта и т. д.;
- ♦ одного файла с расширением `vstemplate`. Этот файл содержит метаданные, которые предоставляют Visual Studio сведения, необходимые для отображения шаблона в диалоговом окне **New Project**.

Эти файлы сжаты в ZIP-файл и находятся в папке `Program Files\Microsoft Visual Studio 10.0\Common7\IDE\ItemTemplatesCache\CSharp`.

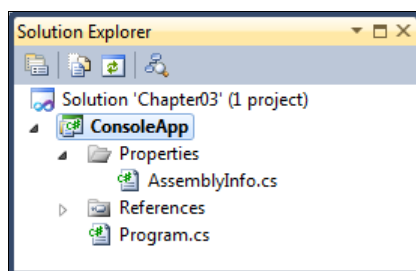
Создание проекта

Вы можете создать новый проект, выбирая **File | New | Project** в меню Visual Studio. При этом откроется диалоговое окно **New Project** с разнообразными видами проектов. Для рассмотрения работы в интегрированной среде разработки мы создадим простое консольное приложение с именем `ConsoleApp` (рис. 3.1).

Консольное приложение позволит ограничиться небольшим объемом кода и сосредоточиться на наиболее важных вопросах изучения работы в среде разработки Visual Studio 2010.

Рис. 3.1. Диалоговое окно **New Project**

На рис. 3.2 показано окно **Solution Explorer**, в котором отображается дерево файлов проекта, файл с кодом `Program.cs` и дополнительный файл `AssemblyInfo.cs` в папке `Properties`.

Рис. 3.2. Дерево проекта в окне **Solution Explorer**

Таков консольный проект, построенный по умолчанию, я не буду подробно рассматривать его содержимое, т. к. вы наверняка уже создавали подобные проекты.

Существует несколько способов откомпилировать и выполнить программу из среды Visual Studio. Например, можно нажать комбинацию клавиш `<Ctrl>+<Shift>+` или выбрать в меню пункт **Build | Build Solution**. В качестве альтернативы можно щелкнуть по кнопке **Build**, расположенной на одноименной панели инструментов.

Чтобы выполнить программу без отладчика, можно нажать комбинацию клавиш **<Ctrl>+<F5>** или выбрать в меню пункт **Debug | Start Without Debugging**.

Программу можно выполнить, не выделяя компиляцию в отдельный этап. В зависимости от выбранных параметров среда IDE сохранит файл, откомпилирует и выполнит его.

Решения

В предыдущем разделе Visual Studio фактически уже создал решение — хотя это решение содержит пока только один проект в окне **Solution Explorer** (см. рис. 3.2), который представляет древовидную структуру, определяющую ваше решение.

Решение может содержать несколько проектов, тогда как проект обычно содержит несколько элементов. Фактически решение — это набор всех проектов, которые составляют пакет создаваемых вами программ.

Окно **Solution Explorer** позволяет вам группировать и управлять множеством файлов, которые составляют ваше приложение. Решение обычно содержит несколько проектов. Проект группирует относящиеся к нему файлы, как показано на рис. 3.3. Например, вы можете создать веб-сайт, приложение Windows Forms, библиотеку классов, консольное приложение и т. д.

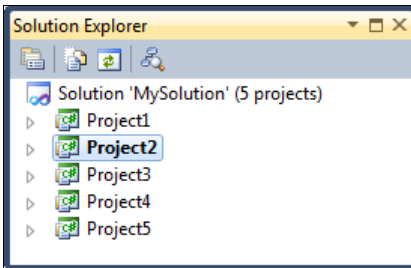


Рис. 3.3. Дерево проектов в окне **Solution Explorer**

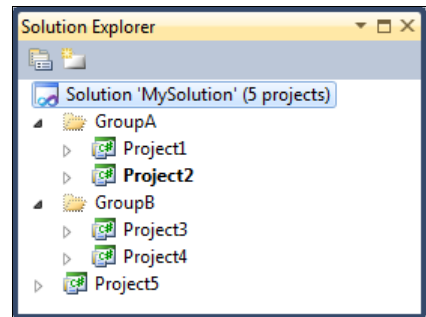


Рис. 3.4. Группировка проектов по папкам

Если решение состоит из множества проектов различного типа, их можно группировать по папкам (рис. 3.4). Папки решения являются средством структуризации в окне обозревателя решений; соответствующие папки Windows не создаются, однако лучше структурировать проекты на диске таким же образом, как и в решении. Для создания папки щелкните правой кнопкой мыши на значке решения, выберите пункт меню **Add | New Solution Folder** и присвойте папке имя.

Добавление проектов в решение

Вы можете создать новый проект двумя способами:

- ◆ выбрать **New Project** в меню **File** (поскольку вы уже сделали);
- ◆ выбрать **Project Add New** в меню **File**.

Если вы выберете команду **Add to solution**, в существующее решение с консольным проектом добавится новый проект, как показано на рис. 3.5.

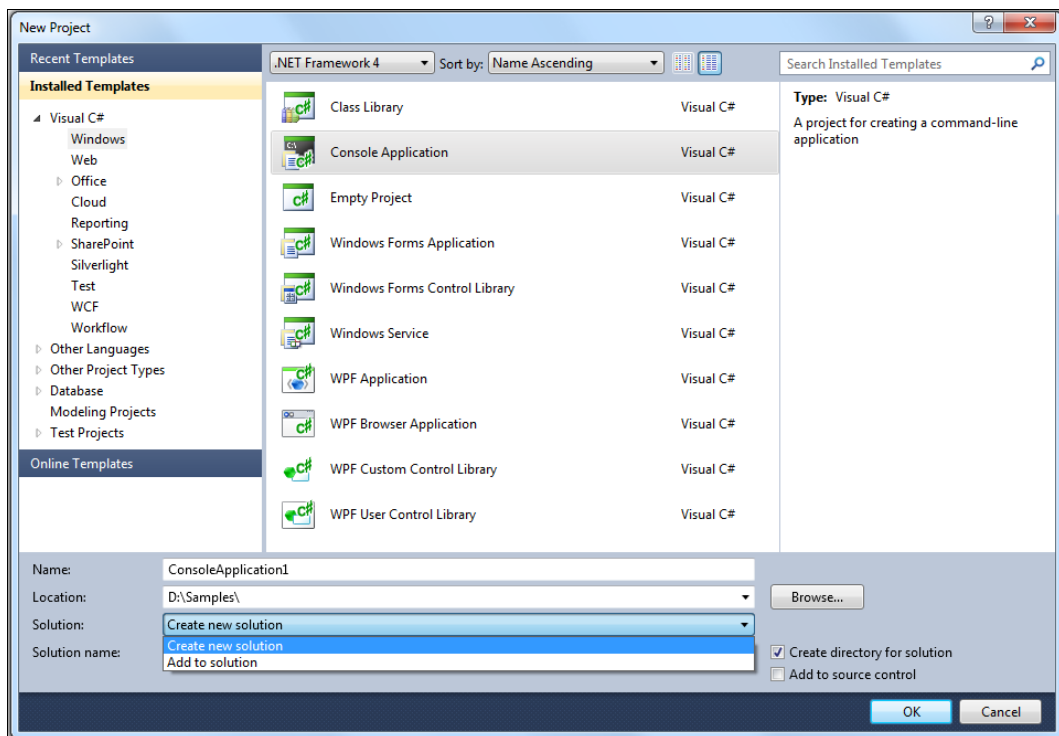


Рис. 3.5. Добавление нового проекта в решение

В соответствии с языковой независимостью решения, в Visual Studio новый проект не обязательно должен быть написан на C#. Допускается возможность размещения проекта C#, проекта Visual Basic и проекта C++ в одном и том же решении. Однако в нашей книге мы будем работать с проектами на языке C# .NET.

Если требуется переименовать какой-нибудь проект или файл проекта, лучше это сделать в окне **Solution Explorer**, т. к. среда разработки Visual Studio автоматически обновит любую ссылку на этот файл в другом проекте. Если же вы переименуете файл через Windows Explorer, Visual Studio не будет в состоянии определить местонахождение этого файла, и тогда потребуются вручную отредактировать проект и решение.

Установка стартового проекта

Когда вы компилируете решение, то компилируются все содержащиеся в нем проекты, но может выполняться только один из них. Поэтому необходимо отдельно указывать среде Visual Studio, какой из проектов должен запускаться на выполнение при нажатии клавиши <F5> или выборе команды **Start**. Если у вас имеется

один исполняемый файл и несколько библиотек, к которым он обращается, то ясно, что запускаться должен именно исполняемый файл. Стартовый проект будет отображаться в дереве решений жирным шрифтом (см. рис. 3.3 и 3.4 ранее в этой главе).

Конфигурации *Debug* и *Release*

При отладке в код добавляются дополнительные строки, обеспечивающие отображение важной отладочной информации. Совершенно очевидно, что было бы предпочтительно полностью удалить эти строки из исполняемого файла, прежде чем включать его в поставляемое программное обеспечение. Такое удаление можно выполнить и вручную, но разве эта задача не упростилась бы намного, если бы имелась возможность каким-то образом просто пометить соответствующие операторы, чтобы компилятор мог игнорировать их при компиляции окончательной версии приложения перед его поставкой?

Компиляция программного продукта на стадии отладки отличается от компиляции окончательной версии. Все, что Visual Studio должна сделать, для того чтобы обеспечить поддержку различных вариантов компоновки проекта, — это сохранить различные наборы вариантов компоновки, которые называются *конфигурациями*. Когда вы создаете проект, Visual Studio автоматически предоставляет вам две конфигурации, соответствующие отладочной (**Debug**) и окончательной (**Release**) версиям продукта:

- ◆ конфигурация **Debug** указывает на то, что оптимизация выполняться не должна, а дополнительная отладочная информация, наоборот, должна включаться в исполняемый код;
- ◆ конфигурация **Release** указывает на то, что оптимизация должна выполняться, а отладочная информация не должна включаться в исполняемый код.

Вам также предоставляется возможность самостоятельно определить параметры конфигурации. Это может потребоваться в тех случаях, когда вы, например, хотите настроить отдельные варианты компоновки проекта, предназначенные для использования на уровне профессионалов или на уровне предприятий, чтобы можно было поставлять две версии данного программного обеспечения.

Выбор конфигурации

В связи с тем, что среда Visual Studio в состоянии хранить подробную информацию о нескольких конфигурациях, возникает очевидный вопрос: каким образом она определяет, какую именно конфигурацию следует использовать для компоновки того или иного проекта? Суть ответа состоит в следующем: всегда существует активная конфигурация, которую Visual Studio и будет применять для компоновки проекта. Также обратите внимание, что конфигурации устанавливаются для каждого проекта в отдельности, а не для решения в целом.

По умолчанию, когда вы создаете проект, в качестве активной устанавливается конфигурация **Debug**. Чтобы установить в качестве активной другую configura-

цию, следует выбрать команду меню **Build | Configuration Manager**. При этом откроется окно свойств решения (рис. 3.6). Доступ к этой команде можно получить также через выпадающее меню основной панели инструментов Visual Studio.

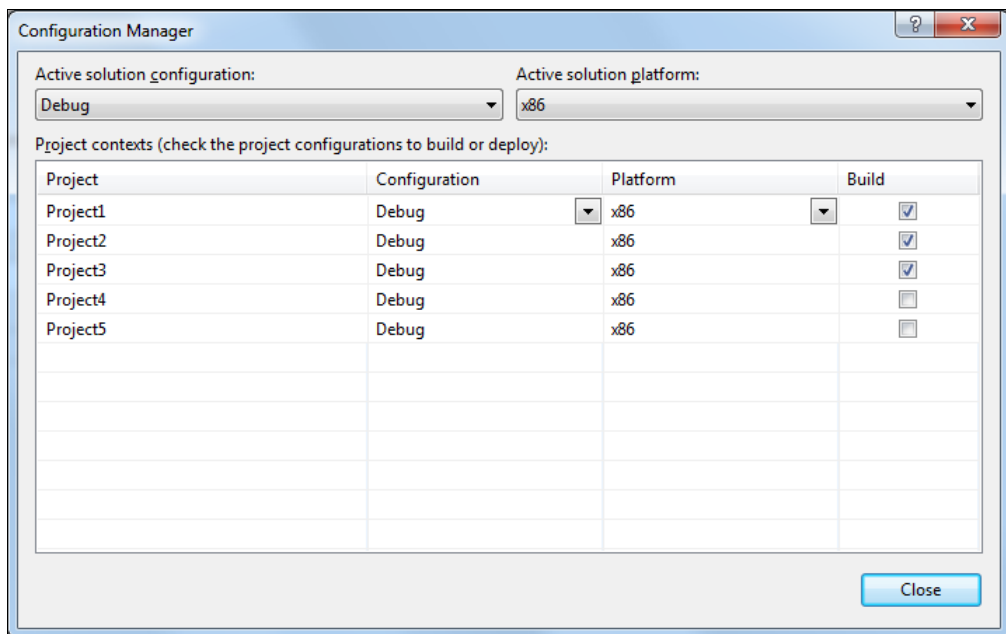


Рис. 3.6. Установка активной конфигурации

Редактирование конфигурации

Помимо выбора активной конфигурации вы также можете просматривать конфигурации и редактировать их. Для этого необходимо выделить соответствующий проект в окне **Solution Explorer**, а затем выбрать команду меню **Projects Properties**. В результате этого на экране отобразится диалоговое окно с весьма сложной структурой. Это же диалоговое окно можно вызвать и иначе, щелкнув правой кнопкой мыши на имени проекта в окне **Solution Explorer** и выбрав в открывшемся контекстном меню команду **Properties**.

В этом диалоговом окне отображается древовидная структура, позволяющая просматривать и редактировать множество общих аспектов проекта. Для полного их обсуждения в данной книге просто не хватило бы места, однако наиболее важные из них мы все же рассмотрим. На рис. 3.7 показан вид вкладки, на которой отображаются доступные свойства для отдельного приложения. При необходимости, например, можно поменять название сборки или пространство имен по умолчанию.

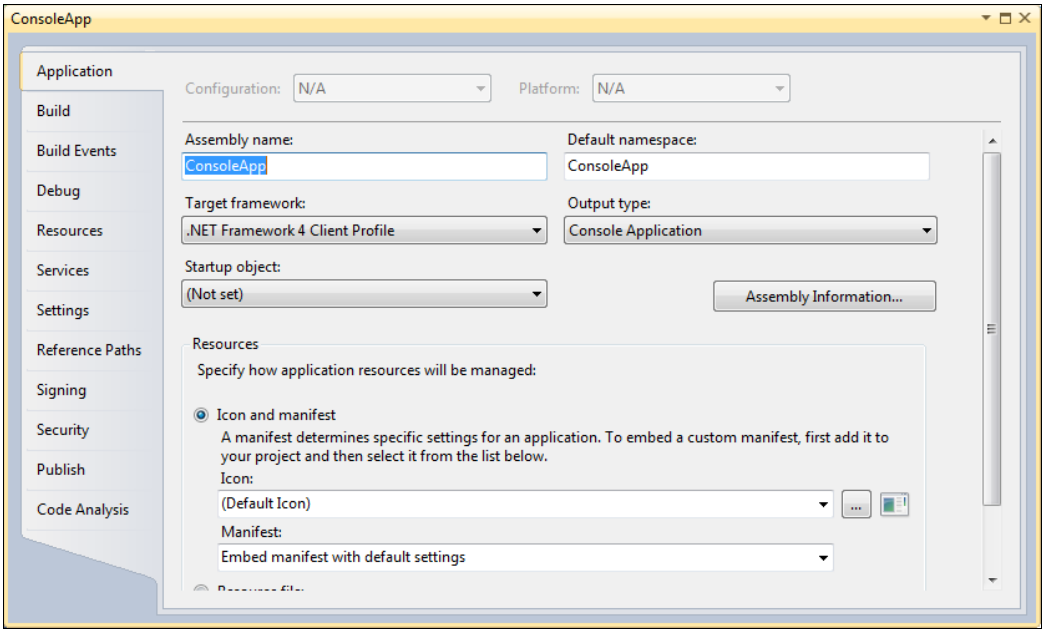


Рис. 3.7. Вкладка **Application** свойств проекта

Рефакторинг кода

Многие программисты разрабатывают свои приложения, кодируя сначала функциональные возможности, а затем они переделывают приложения, чтобы сделать их более управляемыми и более читаемыми. *Рефакторинг*, если кто незнаком с этим термином, является процессом изменения внутренней структуры программы, не затрагивающий ее внешнего поведения и имеющий целью облегчить понимание ее работы.

Поэтому среда Visual Studio 2010 включает ряд инструментальных средств для рефакторинга кода. Вы можете найти эти инструменты под опцией **Refactor** в главном меню Visual Studio. Инструменты для рефакторинга кода значительно упрощают выполнение рефакторизации кода не только в пределах одной страницы, но и по всему приложению. Кроме того, вы также получаете возможность выполнять следующие действия:

- ◆ изменять имена методов, локальных переменных, полей и множества других элементов;
- ◆ извлекать методы из выделенных фрагментов кода;
- ◆ извлекать интерфейсы на основании набора существующих элементов типов;
- ◆ превращать локальные переменные в параметры;
- ◆ переименовывать или переупорядочивать параметры.

В процессе создания учебных программ, описанных в этой книге, а также в дальнейшем, при создании ваших собственных приложений, вы сможете убедиться

в том, что новые средства для рефакторинга кода, предлагаемые средой разработки Visual Studio 2010, обеспечивают прекрасные возможности для того, чтобы сделать код более понятным, удобочитаемым и лучше структурированным.

Система окон

Visual Studio предоставляет множество окон, которые отображают информацию, необходимую для создания приложений. Некоторые окна, такие как **Solution Explorer**, которое мы уже рассмотрели ранее в этой главе, отображены по умолчанию, другие, например отладочные, появляются при запуске приложения в режиме отладки.

Полный список доступных окон расположен в главном меню Visual Studio под опцией **View** (рис. 3.8).

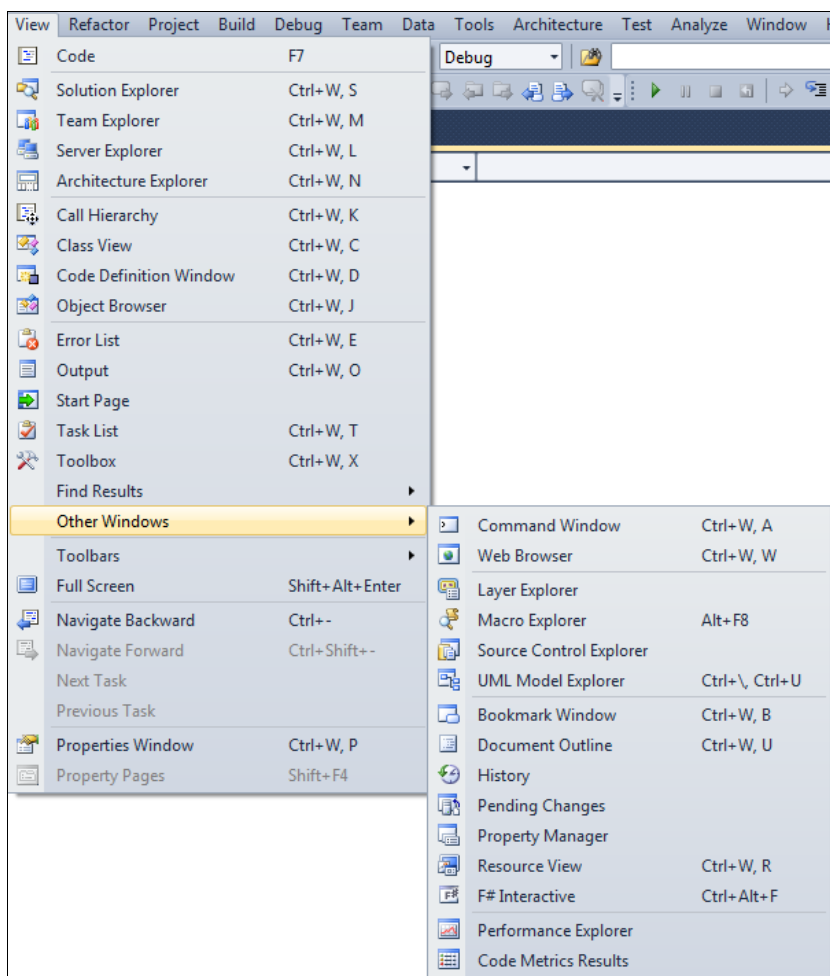


Рис. 3.8. Меню View

В целом, интегрированная среда разработки содержит два типа окон:

- ◆ окна инструментов;
- ◆ окна документов.

Размер области для просмотра и редактирования кода устанавливается в зависимости от размещения окон в интегрированной среде разработки.

Расположение окон

Окна инструментов и документов можно упорядочить перетаскиванием, командами меню **Window**, или щелкнув правой кнопкой мыши заголовок перемещаемого окна. Панели инструментов можно перетаскивать или упорядочивать с помощью диалогового окна **Customize** (Настройка), доступ к которому можно получить через команду меню **Tools | Customize**.

Любое окно инструментов или окно документов может быть отстыковано от интегрированной среды разработки и помещено в любое место на рабочем столе. Если два связанных окна документов отображаются одновременно, при изменении содержимого в любом из них обновляются оба этих окна.

Окна инструментов можно прикрепить к одной из сторон фрейма интерфейса IDE. При перетаскивании окна инструментов в новое расположение в интегрированной среде разработки появляется маркер в виде ромба. Маркер помогает закрепить окно инструментов на одной из четырех сторон интегрированной среды разработки или во фрейме редактирования.

Чтобы переместить закрепляемое окно без его привязки к какому-либо месту, нужно при его перетаскивании держать нажатой клавишу <Ctrl>.

На рис. 3.9 изображены специальные маркеры, которые появляются при перетаскивании окна инструментов или документов к центру интегрированной среды разработки.

При перетаскивании окон появляется маркер в виде ромба. Четыре стрелки ромба указывают на четыре стороны панели редактирования. Если окно является окном инструментов, то дополнительные четыре стрелки указывают на углы окна IDE.

Когда перетаскиваемое окно достигнет нужного расположения, наведите указатель на соответствующую часть ромба-маркера. Указанная область будет отображена затемненной. Чтобы закрепить окно, отпустите кнопку мыши, и оно останется прикрепленным в выбранном месте среды разработки.

Например, если обозреватель решений закреплен на правой стороне среды разработки, а вы хотите закрепить его на левой стороне, перетащите обозреватель решений в центр среды разработки, наведите указатель на самую левую стрелку ромба и отпустите кнопку мыши.

Кроме того, окно инструментов можно прикрепить к боковой, верхней или нижней части окна среды разработки, перетащив его в сторону до появления второго маркера в форме ромба. Щелкните одну из четырех стрелок, чтобы закрепить окно рядом с данной частью боковой стороны окна.

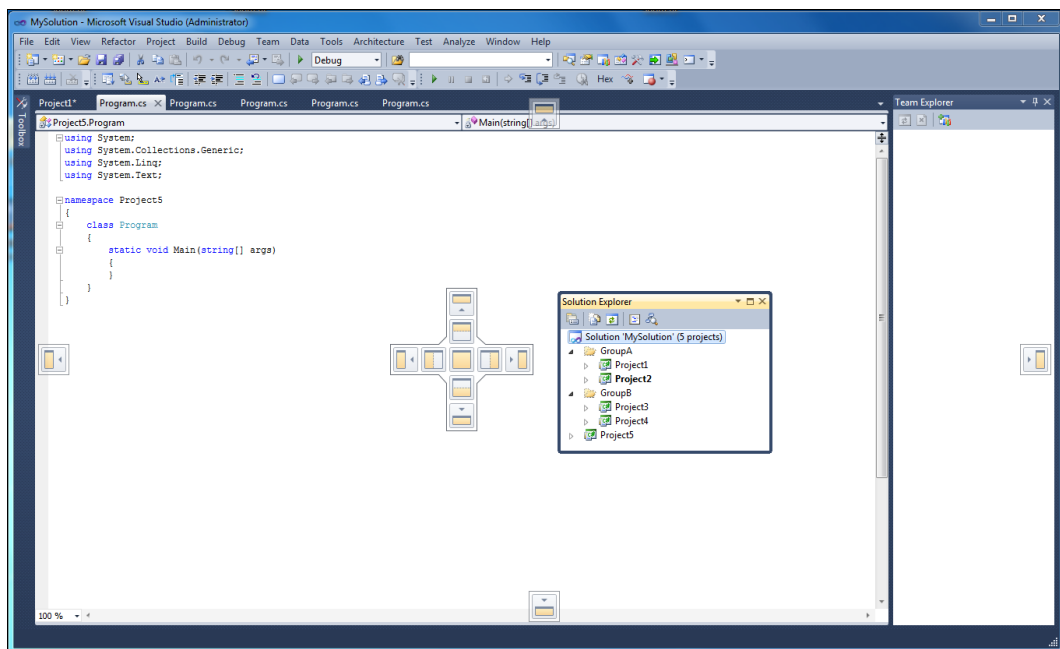


Рис. 3.9. Перетаскивание и закрепление окна

Все окна инструментов, названия которых встречаются в меню **View**, поддерживают возможность *автоматического скрывтия* (значок канцелярской кнопки в заголовке окна). Автоматическое скрывтие сдвигает окно в сторону, когда активно другое окно. Когда окно скрыто, его имя и значок отображаются на вкладке на краю интегрированной среды разработки. Чтобы снова сделать окно активным, наведите указатель на вкладку, и оно вернется в поле зрения.

Окно *Properties*

Окно свойств **Properties** — основной инструмент настройки формы и ее компонентов. Содержимое этого окна представляет собой весь список свойств выбранного в данный момент компонента или формы. Вызывается это окно несколькими способами:

- ◆ в меню **View** выбираем пункт **Properties Window** (или используем клавишу <F4>);
- ◆ на выбранном объекте щелкаем правой кнопкой мыши и в контекстном меню находим пункт **Properties**;
- ◆ выбираем объект и нажимаем клавишу <F4>;
- ◆ просто выбираем объект и переходим в окно **Properties**.

При создании проекта в окне **Properties** отображаются свойства проекта. Кроме того, окно **Properties** позволяет управлять размером, внешним видом и поведением создаваемых элементов управления.

Окно **Properties** также группирует схожие свойства в наборы (для облегчения доступа). На рис. 3.10 показано окно **Properties**.

Обратите внимание, что по умолчанию это окно группирует схожие свойства в разделы при помощи категорий **Advanced**, **Misc** и др. При желании можно отключить группировку свойств по категориям и отсортировать список свойств в алфавитном порядке путем нажатия значка **AZ** на панели инструментов. Наконец, окно **Properties** позволяет привязать события элемента управления к коду внутри нашего приложения.

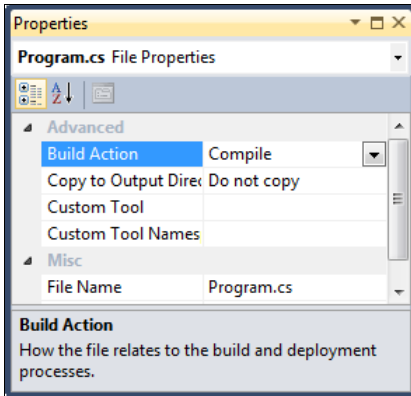


Рис. 3.10. Окно **Properties**

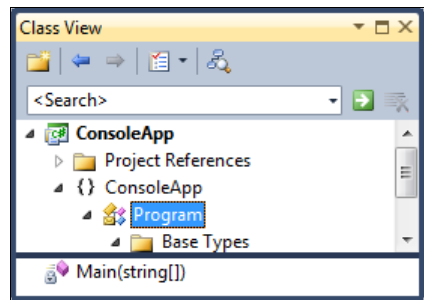


Рис. 3.11. Окно **Class View**

Окно **Class View**

Окно просмотра структуры классов **Class View** позволяет перемещаться в коде по выбранному объекту и содержит методы, классы, данные всего листинга проекта.

В Visual Studio окно **Class View** в действительности выступает не как самостоятельное окно, а как вкладка окна **Solution Explorer**. По умолчанию окно **Class View** даже не появляется в окне **Solution Explorer**. Чтобы отобразить его на экране, выберите в главном меню пункт **View | Class View** или воспользуйтесь сочетанием клавиш **<Ctrl>+<Shift>+<C>**.

В окне **Class View** (рис. 3.11) отображается иерархия пространств имен и классов, присутствующих в вашем коде, в виде дерева, которую вы можете развернуть для получения более подробной информации о том, какие классы содержатся в пространствах имен и какие элементы содержатся в классах.

Удобной возможностью, предлагаемой окном **Class View**, является то, что если вы щелкнете правой кнопкой мыши на имени любого элемента, к исходному коду которого у вас имеется доступ, то в открывшемся контекстном меню будет присутствовать команда **Go To Definition** (Перейти к определению), позволяющая перейти к тому месту программы в окне редактора кода, в котором находится определение данного элемента. Того же результата можно добиться, дважды щелкнув на элементе в окне **Class View** или щелкнув правой кнопкой мыши на названии эле-

мента в редакторе исходного кода и выбрав ту же команду в открывшемся контекстном меню.

Кроме того, контекстное меню позволяет добавить в класс поле, метод, свойство или индексатор. Это означает, что вы устанавливаете детальные характеристики соответствующего элемента в диалоговом окне, и для вас автоматически генерируется соответствующий код.

Окно *Object Browser*

В процессе написания кода часто требуется информация о том, какие методы и другие элементы программного кода доступны в базовых классах и других библиотеках, на которые в ваших сборках имеются ссылки. Для этого предназначено окно браузера объектов **Object Browser**. В Visual Studio 2010 для доступа к этому окну надо выбрать в главном меню пункт **View | Object Browser**.

Окно **Object Browser** по своему внешнему виду напоминает окно **Class View**. Окно **Object Browser** тоже отображает древовидное представление структуры классов вашего приложения, позволяя просматривать члены каждого класса. Незначительное отличие пользовательского интерфейса состоит в том, что члены класса отображаются на отдельной панели, а не в дереве класса. Действительно важным отличием является то, что с помощью этого окна вы можете просмотреть не только пространства имен и классы, входящие в состав вашего проекта, но и все сборки, на которые в вашем проекте имеются ссылки. На рис. 3.12 представлен вид окна **Object Browser** при просмотре класса `Console`, входящего в состав базовых классов .NET.

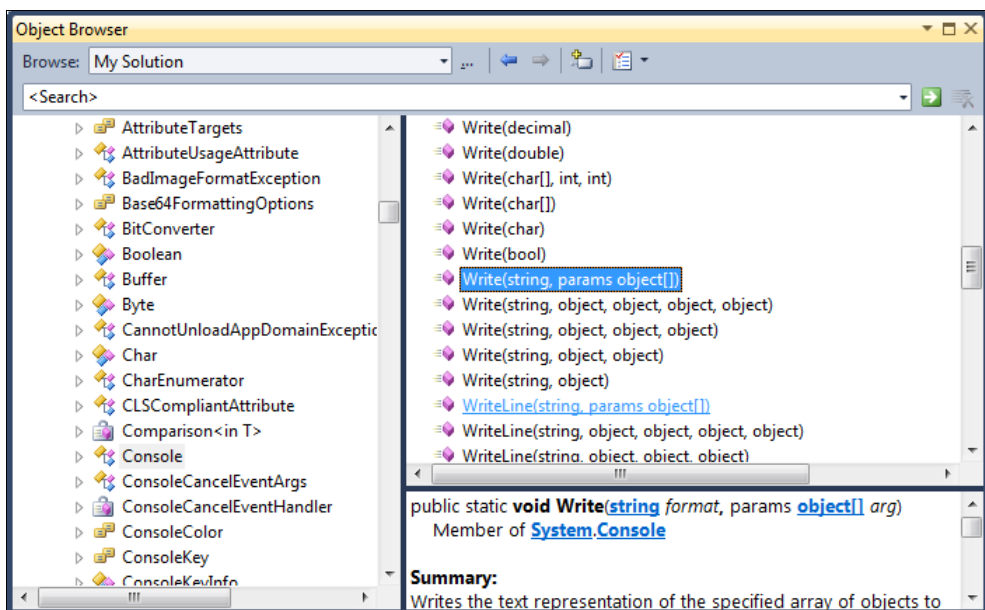


Рис. 3.12. Окно **Object Browser**

При работе с окном **Object Browser** необходимо учитывать, что классы в нем группируются сначала по сборкам, в которых они находятся, и лишь затем по пространствам имен. К сожалению, поскольку пространства имен для классов часто разбросаны по нескольким сборкам, нахождение определенного класса может оказаться затруднительным, если вы не знаете, какой сборке он принадлежит.

Окно **Code Definition**

В окне **Code Definition** (Определение кода) выводится исходный код для объекта или элемента. Если исходный код для этого элемента недоступен, то интегрированная среда разработки отображает метаданные в виде исходного кода в окне **Code Definition**.

Например, если в редакторе расположить курсор в пределах слова `Console`, то метаданные типа `Console` будут показаны в виде исходного кода в окне **Code Definition**. Исходный код напоминает объявление класса `Console`, но не содержит его реализации. Внешний вид окна **Code Definition** показан на рис. 3.13.

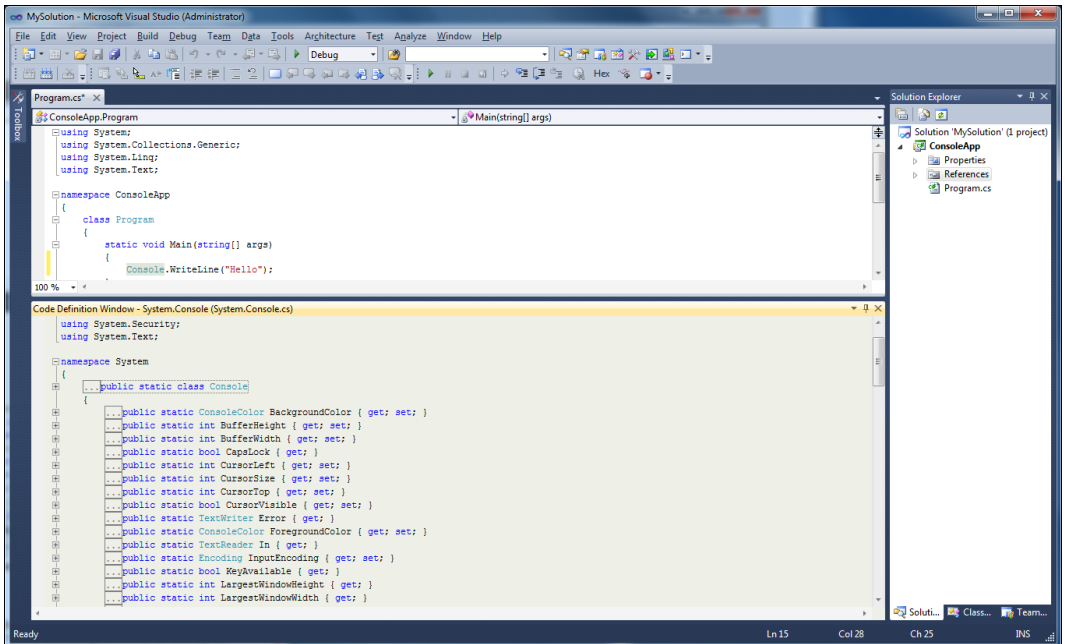


Рис. 3.13. Окно **Code Definition**

Если требуется посмотреть объявление элемента, который присутствует в окне **Code Definition**, щелкните правой кнопкой мыши этот элемент и выберите команду **Goto definition**.