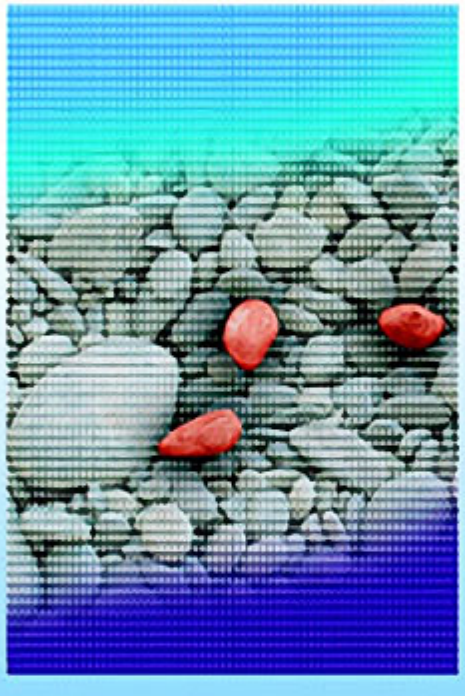
Алексей Голощапов

bhv

# 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

**Голощапов 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 | *Екатерина Кондукова* |
| Зам. главного редактора | *Игорь Шишигин* |
| Зав. редакцией | *Григорий Добин* |
| Редактор | *Екатерина Капалыгина* |
| Компьютерная верстка | *Ольги Сергиенко* |
| Корректор | *Зинаида Дмитриева* |
| Дизайн серии | *Инны Тачиной* |
| Оформление обложки | *Елены Беляевой* |
| Зав. производством | *Николай Тверских* |

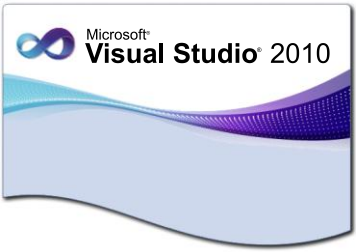
Лицензия ИД № 02429 от 24.07.00. Подписано в печать 31.01.11.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 43,86.
Тираж 1500 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

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

# Оглавление




# 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.

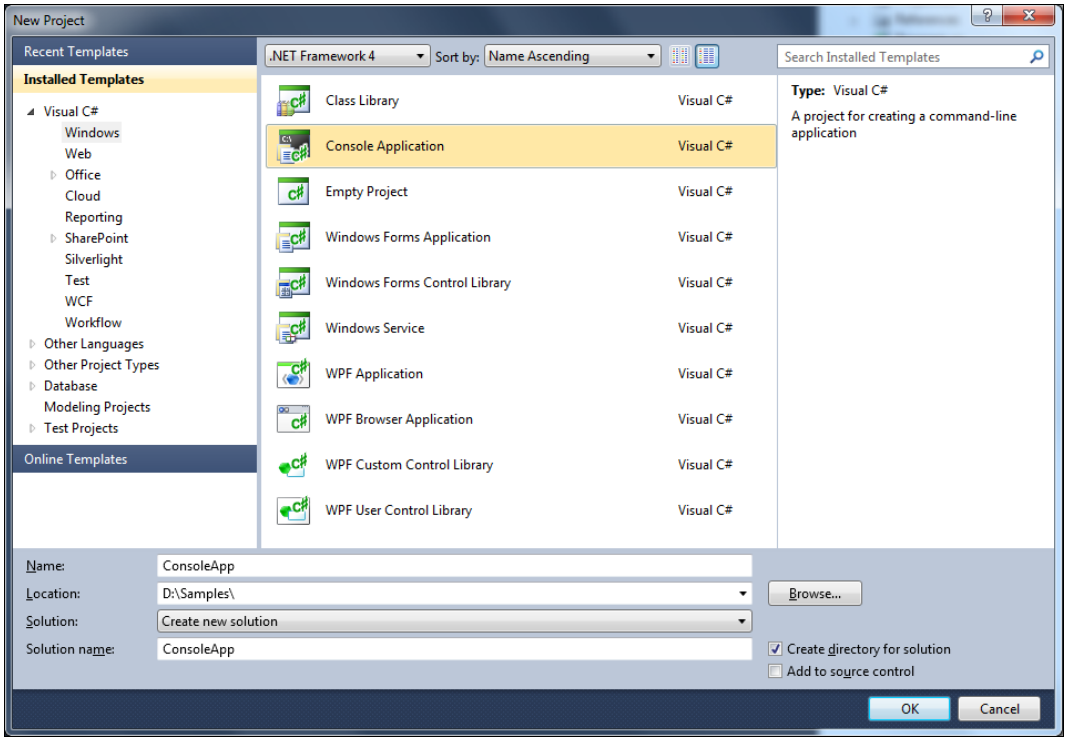



**Рис. 3.1.** Диалоговое окно **New Project**

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.

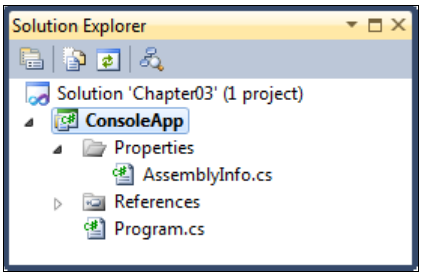



**Рис. 3.2.** Дерево проекта в окне **Solution Explorer**

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>+<B> 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.

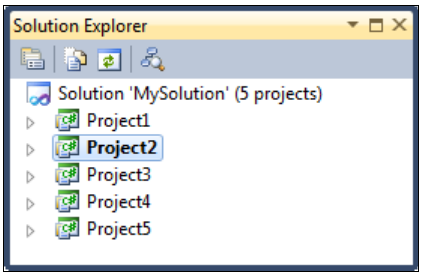



**Рис. 3.3.** Дерево проектов в окне **Solution Explorer**

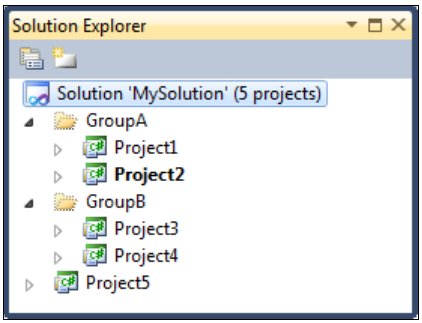



**Рис. 3.4.** Группировка проектов по папкам

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.

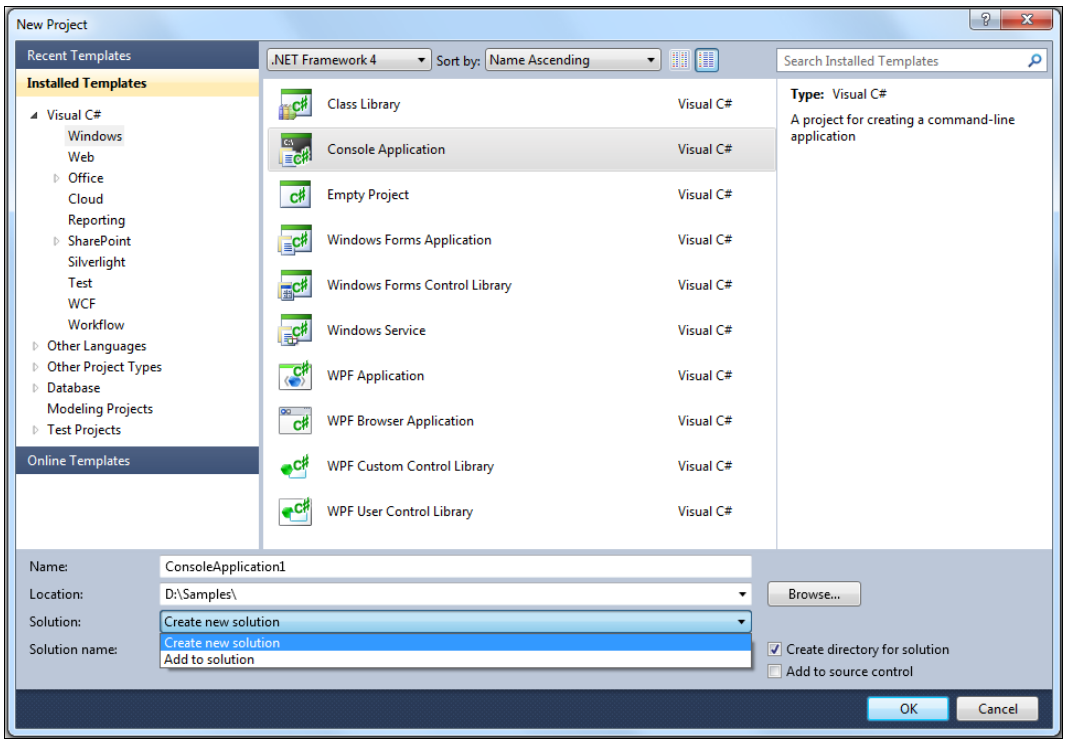



**Рис. 3.5.** Добавление нового проекта в решение

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 потребуется 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 конфигура-

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.

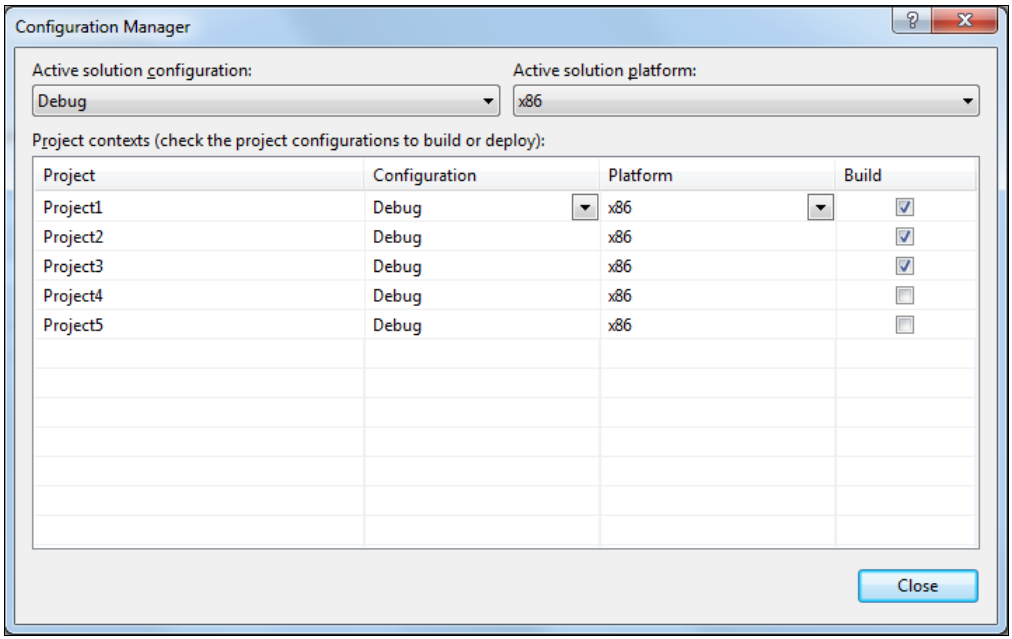



**Рис. 3.6.** Установка активной конфигурации

## 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.

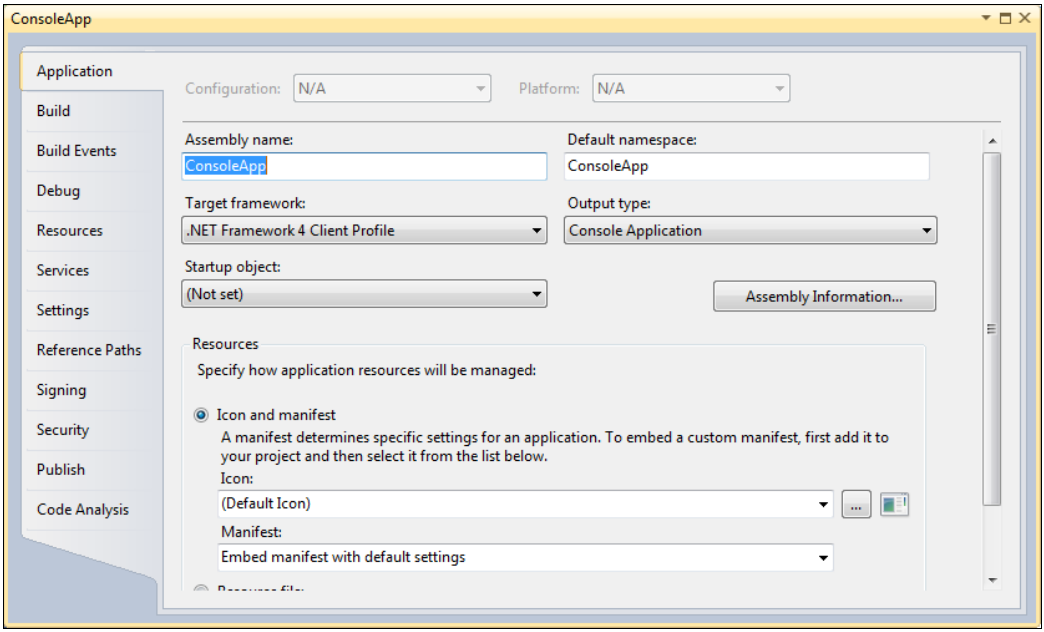



**Рис. 3.7.** Вкладка **Application** свойств проекта

## 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).

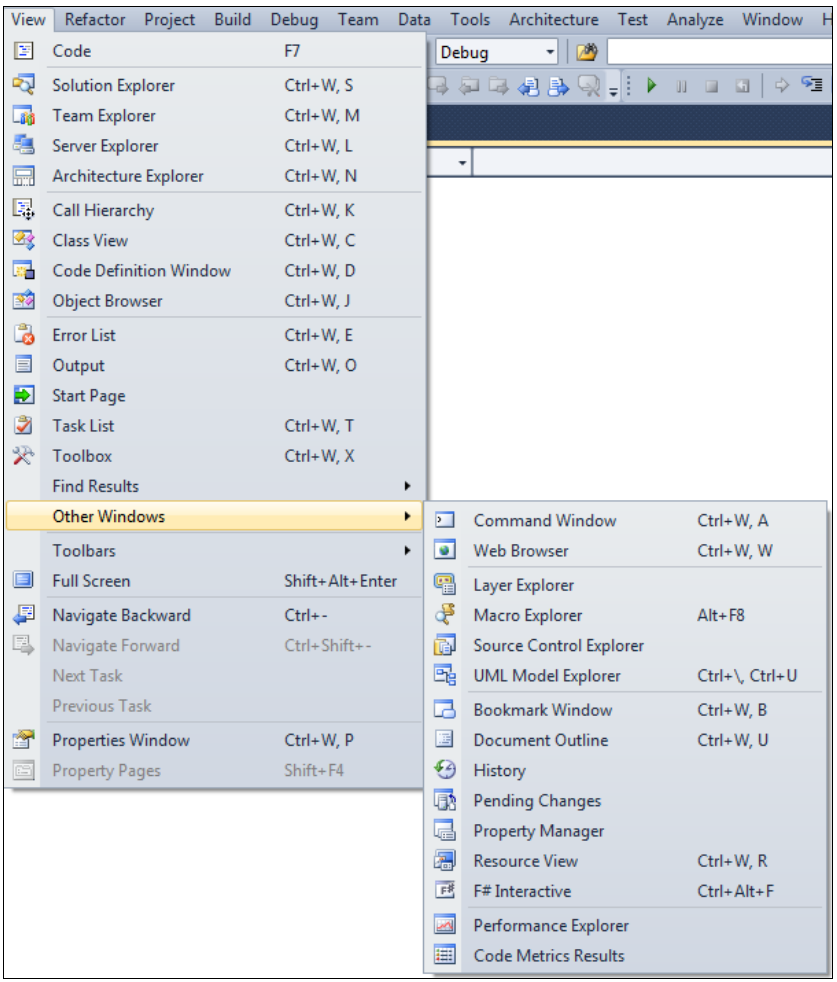



**Рис. 3.8.** Меню **View**

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.

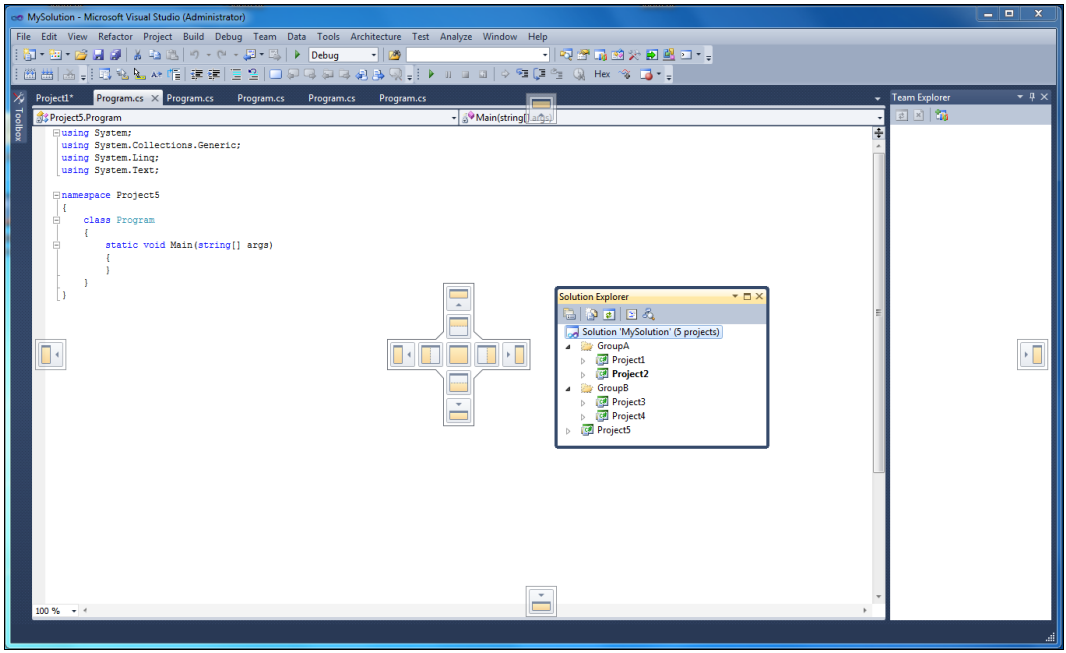



**Рис. 3.9.** Перетаскивание и закрепление окна

Все окна инструментов, названия которых встречаются в меню **View**, поддерживают возможность *автоматического скрытия* (значок канцелярской кнопки в заголовке окна). Автоматическое скрытие 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.

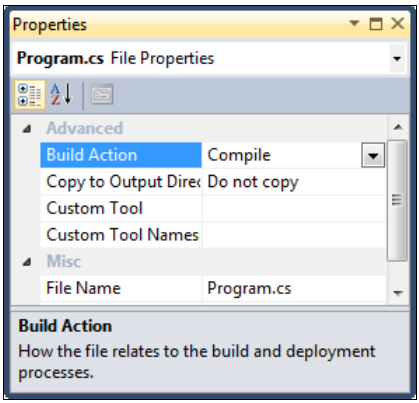



**Рис. 3.10.** Окно **Properties**

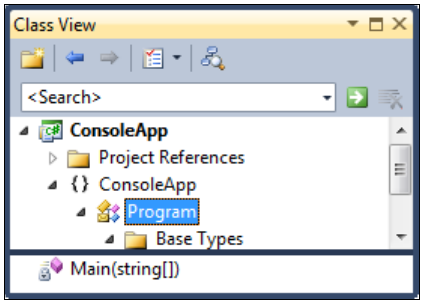



**Рис. 3.11.** Окно **Class View**

## 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.

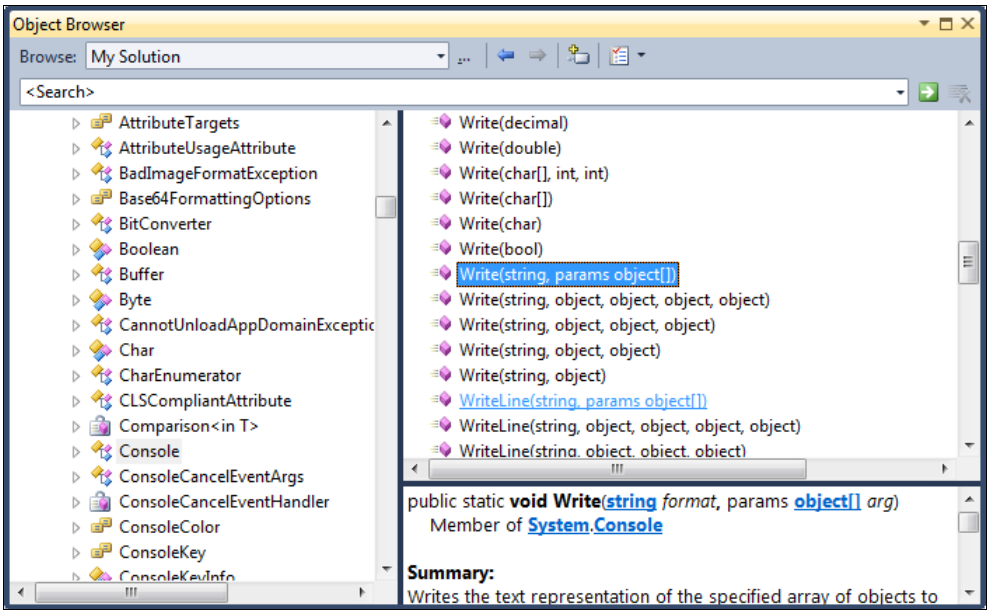



**Рис. 3.12.** Окно **Object Browser**

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.

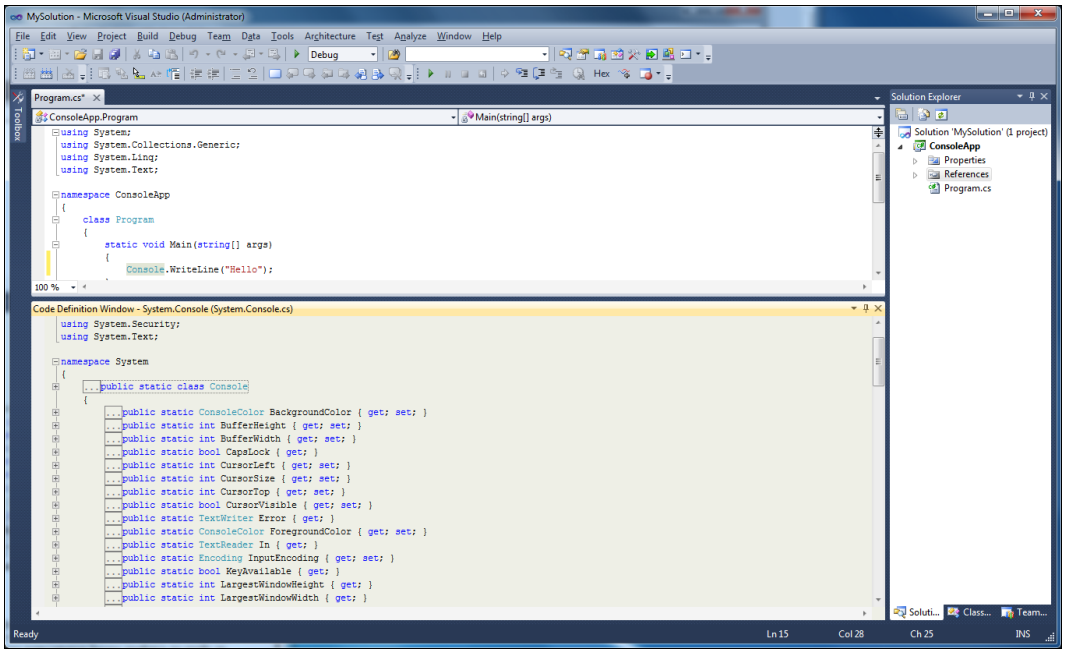



**Рис. 3.13.** Окно **Code Definition**

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