陈 锵 孙赫雄 陈 楠 编著
邱仲潘 审校

Visual C++ 2005 从入门到精通（普及版）

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Visual C++ 2005 从入门到精通

（普及版）

陈　锵　孙赫雄　陈　楠　编著

邱仲潘　审校

電子工業出版社

Publishing House of Electronics Industry

北京 • BEIJING

内 容 简 介

本书全面介绍了 Visual C++ 2005 的特点、使用方法及编程技巧，旨在提供 Visual C++的“从入门到精通”式的综合性指南。其内容包括 Visual C++集成式编程环境，Visual Studio、Windows GUI 编程，微软基础类、应用程序向导、类向导、类库和 ActiveX 控件的使用，以及文件访问和图形打印等。

本书为“从入门到精通”类图书，适合阅读的读者范围涵盖初学者到高级技术人员。

图书在版编目(CIP)数据

Visual C++ 2005 从入门到精通 （普及版）/ 陈锵，孙赫雄，陈楠编著.—北京：电子工业出版社，2007.5
ISBN 978-7-121-04025-2

I. V… II.①陈…②孙…③陈… III.C 语言－程序设计 IV.TP312

中国版本图书馆 CIP 数据核字（2007）第 037194 号

责任编辑：王军花
印　　刷：北京天竺颖华印刷厂
装　　订：三河市金马印装有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编：100036
　　北京市海淀区翠微东里甲 2 号　邮编：100036
开　　本：787×1092　1/16　印张：26.625　字数：680 千字
印　　次：2007 年 5 月第 1 次印刷
定　　价：40.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系。联系电话：（010）68279077；邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

Visual C++ 2005 是 Microsoft 公司开发的基于 C/C++的集成开发工具，是一种公认的功能强大的、效率高的开发工具。Visual C++ 2005 在原有的基础上增加了新特性，不仅支持原来的 ISO/ANSI 标准 C++，而且还支持名为 C++/CLI 的新版 C++。本书将讲述使用 C++/CLI 新特性的优越性和使用 C++开发应用程序的基础知识。

本书全面介绍了 Visual C++ 2005 的开发环境，详细阐述了 C++的语言基础以及面向对象编程的概念和方法，各章按照由浅入深、步步为营的方法进行讲述。书中包含大量的实例，通过对这些实例进行剖析，力求把一些编程方法和技巧形象化地呈献给读者。

本书对如何利用 Visual Studio 进行程序开发进行了全面细致的介绍。虽然 ISO/ANSI C++仍然是许多专业人员的最爱，但 C++/CLI 以它独特的优越性使得它也成了基本的语言。因此，本书中包括这两种 C++语言的基本内容，真正使读者朋友对 Visual C++ 2005 做到从入门到精通。

本书适合的读者对象：

1．从来没接触到 C/C++，但十分渴望能马上加入 C++程序员的行列，体验美妙的编程乐趣的人。

2．以前接触过其他语言，具备少量的编程经验，希望通过学习 C++，提升实际的 Microsoft Windows 编程技能的人。

3．有一些使用 C/C++编程的经验，但编程环境是在非 Microsoft Windows 下，希望通过使用 Visual C++ 2005，积累在 Windows 环境下编程技能的人。

本书作者在编写过程中倾心相注、精心而为。但由于时间仓促、水平有限，难免有疏忽和纰漏之处，欢迎广大读者批评指正，提出宝贵的意见和建议，你们的支持是我们最大的财富。

本书由陈锵、孙赫雄、陈楠编著，邱仲潘审校。此外，陈初铨、宫正、汪少明也参与了本书的编写工作，另外，刘文红、刘文琼、温连英提供了多方面的帮助，在此一并表示感谢。

为方便读者阅读，若需要本书配套资料，请登录“华信教育资源网”（http://www.hxedu.com.cn），在“资源下载”频道的“图书资源”栏目下载。

目　录

第 1 章　Visual Studio 开发环境

1.1　Visual C++ 2005 增加的新特性

1.1.1　C++/CLI

在 VS.NET 7.0 推出时，很多 C++程序员带着复杂的情绪接受了这个现实，一方面很高兴能够继续使用C++，另一方面讨厌使用C++的托管扩展语法并且认为其中的错误非常多。Visual C++.NET 2003 中有 98%的部分与 ISO C++标准保持一致，这使它比以往任何版本更靠近这些标准，而且它还加入了对一些功能（如局部模板专用化）的语言支持。

Visual C++ 2005 为.NET 开发提供了既优雅又强大的新语法支持。它使用的新优化技术已经使 Microsoft 产品的运行速度提高了 30%。它通过新的编译模式来确保 Microsoft.NET 框架通用语言基础结构（Common Language Infrastructure，CLI）的一致性和可验证性，并且具有新的 interop 模型，这不仅提供了本机和托管环境的无缝合并，而且还在跨这些边界的情况下提供了完全控制。

VS.NET 7.0 旧语法的问题如下：

- “丑陋”的语法和文法：在托管扩展中以双下划线关键字定义垃圾回收类、属性等。
- C++和.NET 的整合度很差，不能在 CLI 上使用 C++的特性，也不能在 C++的类型上使用垃圾回收机制。
- 混乱的指针使用。非托管的 C++指针和托管的引用指针都使用相同的符号“*”，这样很容易混淆，因为自动垃圾回收的指针和非托管的指针在状态和行为上都有很大的不同。
- 编译器不能生成具有可验证性的代码。

C++/CLI（the ISO standard C++ programming language and Common Language Infrastructure）的优点如下：

- 确保编程人员非常自然地编写 C++代码，而且通过对 ISO C++标准的纯粹扩展来提供一种优雅的语法。
- 把.NET 的全部强大功能带给 C++，而与此同时也把 C++的强大功能带给.NET。
- 该编译器增强了前两个版本中提供的缓冲区安全检查选项，并且还包括了 C++应用程序普遍使用的以安全性为中心的库的新版本。

1.1.2　Hello World

下面，我们以“Hello World”为例，向读者展示 C++/CLI 的卓越特性。

```
using namespace System;
```

```
void _tmain()
{
    Console::WriteLine(L"Hello World");
}
```

上述代码看起来和旧的语法差不多。注意这个时候不要增加对 mscorlib.dll 的引用，因为如果引用这个 dll，编译器会默认使用/clr，而且使用的是新语法。

1. 句柄（Handles）

在旧的语法中，一个容易引起混淆的地方是对非托管的指针和托管的引用都使用“*”符号，所以在 C++/CLI 中引入了句柄的概念，下面举例进行说明。

```
void _tmain()
{
    //^表示一个句柄
    String^ str = "Hello World";
    Console::WriteLine(str);
}
```

“^”表示一个托管对象的句柄。根据 CLI 规范，句柄是对一个托管对象的引用。这样，句柄和指针就不会混淆了。

2. 句柄和指针的不同点

- 指针使用“*”表示；句柄使用“^”表示。
- 句柄是对位于托管堆中的对象的引用；指针则是指向一个内存地址。
- 指针是稳定的，并且内存自动回收机制对它不起作用；内存自动回收机制则对句柄进行自动回收。
- 对指针来说，程序员必须明确地删除，否则会引起内存泄露；对句柄的删除则是可选的。
- 句柄是类型安全的，而指针不是。无法将句柄定义为 void^。
- 用“new”关键字声明返回一个指针；用“gcnew”关键字声明返回一个句柄。

3. CLR（Common Language Runtime）对象范例

CLR 对象范例如下所示：

```
void _tmain()
{
    String^ str = gcnew String("Hello World");
    Object^ o1 = gcnew Object();
    Console::WriteLine(str);
}
```

“gcnew”关键字用来实例化一个 CLR 对象，并且返回一个该对象在 CLR 堆中的引用。“gcnew”最大的好处就是让程序员很容易区分托管和非托管的实例对象。

基本上，使用“gcnew”关键字和“^”操作符就能够满足大部分的使用要求，但是需要自己创建和声明自定义的托管类和接口。

4. 声明类型

CLR 类型都是用加形容词前缀来描述。下面就是几个在 C++/CLI 中的类型声明：

- CLR 类型
  - 引用类型
    - ref class RefClass{…};
    - ref struct RefClass{…};
  - 值类型
    - value class ValClass{…};
    - value struct ValClass{…};
  - 接口
    - interface class IType{…};
    - interface struct IType{…};
  - 枚举
    - enum class Color{…};
    - enum struct Color{…};
- 原生类型
  - class Native{…};
  - struct Native{…};

下面举例说明 CLR 类型的声明：

```
using namespace System;
interface class IDog
{
    void Bark();
};
ref class Dog : IDog
{
public:
    void Bark()
    {
        Console::WriteLine("Bow wow wow");
    }
};
void _tmain()
```

```
{
    Dog^ d = gcnew Dog();
    d->Bark();
}
```

在旧的语法中，需要用到很多带有双下划线的关键字，比如_gc 和_interface，但新的语法不会这样，因此新语法更加简洁易用。

1.2 MFC 与 SDK 之间的关系

SDK 就是 Software Development Kit 的缩写，中文意思就是“软件开发工具包”。这是一个覆盖面相当广的名词，可以这么说：辅助开发某一类软件的相关文档、范例和工具的集合都可以叫做 SDK。由于 SDK 包含了使用 API 的必需资料，所以人们也常把仅使用 API 来编写 Windows 应用程序的开发方式叫做“SDK 编程”。而 API 和 SDK 是开发 Windows 应用程序所必需的东西，所以其他编程框架和类库都是建立在它们之上的，比如 VCL 和 MFC，虽然他们比“SDK 编程”有更高的抽象度，但这丝毫不影响它们在需要的时候随时调用 API 函数。

MFC（Microsoft Foundation Class Library）中的各种类结合起来构成了一个应用程序框架，它的目的就是让程序员在此基础上建立 Windows 下的应用程序，这是一种相对 SDK 来说更为简单的方法。MFC 框架定义了应用程序的轮廓，并提供了用户接口的标准实现方法，因此程序员所要做的就是通过预定义的接口把具体应用程序特有的东西填入这个轮廓。Microsoft Visual C++提供了相应的工具来完成这个工作：AppWizard 可以用来生成初步的框架文件（代码和资源等）；资源编辑器用于帮助直观地设计用户接口；ClassWizard 用来协助添加代码到框架文件；最后进行编译。

MFC 实现了对应用程序的封装，把类、类的继承、动态约束、类的关系和相互作用等封装起来。这样封装的结果对程序员来说，是一套开发模板（或者说模式）。针对不同的应用，程序员采用不同的模板。例如，SDI 应用程序的模板，MDI 应用程序的模板，规则 DLL 应用程序的模板，扩展 DLL 应用程序的模板，OLE/ActiveX 应用程序的模板，等等。

MFC 提供了一个 Windows 应用程序开发模式，其中对程序的控制主要是由 MFC 框架完成的。框架和其本身的处理事件不依赖程序员的代码，但是可以调用程序员的代码来处理应用程序特定的事件。MFC 是 C++类库，程序员就是通过使用继承和扩展适当的类来实现特定的目的的。例如，继承时，应用程序特定的事件由程序员的派生类来处理，不感兴趣的由基类来处理。实现这种功能的基础是 C++对继承的支持、对虚拟函数的支持以及 MFC 实现的消息映射机制。

1.3 .NET Framework 介绍

1.3.1 .NET Framework

.NET Framework 是.NET 在技术层面上的核心。它指的是支持生成和运行下一代应用程序和 XML Web 服务的内部 Windows 组件。

.NET Framework 旨在实现下列目标：

- 提供一个一致的面向对象的编程环境，而不管对象代码是在本地存储、执行，还是在本地执行但在 Internet 上分布，或者是在远程执行。
- 提供一个将软件部署和版本控制冲突最小化的代码执行环境。
- 提供一个可提高代码（包括由未知的或不完全受信任的第三方创建的代码）执行安全性的代码执行环境。
- 提供一个可消除脚本环境或解释环境的性能问题的代码执行环境。
- 使开发人员可以非常容易解决类型不同的应用程序（如基于 Windows 的应用程序和基于 Web 的应用程序）。
- 按照工业标准生成所有通信，以确保基于.NET Framework 的代码可以与任何其他代码集成。

从这些目标可以看出，.NET Framework 旨在构建一个统一的编程环境，减少软件部署和版本控制冲突，提高代码的安全性，使不同语言的代码（都是基于.NET Framework 的）都能完美集成。

.NET Framework 包含两个主要的部分：

一部分是公共语言运行库（Common Language Runtime, CLR），这是我们下一节要讲的内容。如果在本书中提到公共语言运行时，也是指 CLR。

另一部分是一些统一的、分层次的类库——.NET Framework 类库，其中包括：

- ASP.NET——对 ASP 革命性的提升。
- Windows Forms ——用来创建智能客户端的环境，不能简单地理解为 Form 窗体。
- ADO.NET ——一个松藕合的数据库访问子系统。

.NET Framework 类库是一个综合性的面向对象的可重用类型集合，我们可以使用它开发多种应用程序，这些应用程序包括传统的命令行或图形用户界面（GUI）应用程序，也包括基于 ASP.NET 所提供的最新的应用程序（如 Web 窗体和 XML Web 服务）。

在.NET 框架所包含的两个部分之中，公共语言运行库是.NET Framework 类库的基础。

图 1.1 是整个.NET Framework 的环境示意图。

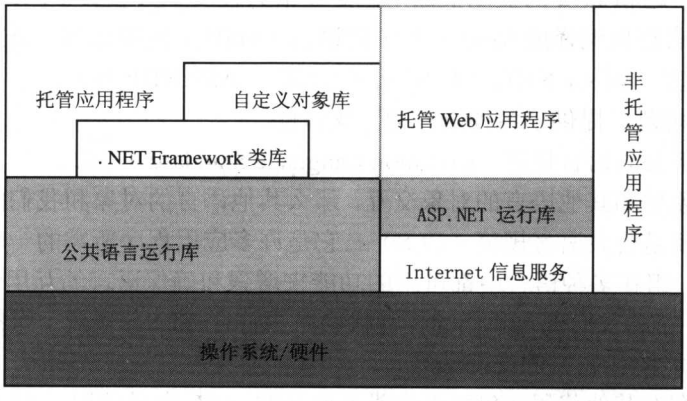

图 1.1　.NET Framework 的环境示意图

图 1.1 仅是一个层次示意图，方框大小并不表示比重。从中不难看出，Internet 在整个.NET 体系中是重要的一环。Internet 基于 ASP.NET 运行库，正如操作系统/硬件基于公共语言运行库那样。

由于.NET Framework 在硬件之上多了一层（运行库和.NET Framework 类库），这为.NET Framework 平台无关性提供了可能。比如目前在 Linux 下有一个叫 Mono 的优秀开源项目，提供了一个 Linux 平台下的“.NET Framework”，也就是说，读者朋友在 Windows 下用 Visual Studio 编译的程序可以在 Linux 下通过 Mono 运行。这有点像用 Wine（Linux 下的另一个开源项目）运行 Win32，但是本质却不一样。不过，目前 Mono 好像只支持.NET Framework 1.0. 而 Visual Studio 2005 默认支持的是.NET Framework 2.0。从架构上来说，.NET Framework 有点像 Java 的虚拟机架构。

读者朋友也许听说过或看过.NET Framework SDK，那么它是什么意思呢？它就是.NET Framework 开发工具包的意思。

它包括：

- 我们以上提到的.NET Framework 的两部分；
- 文档，例子；
- 命令行工具和编译器。

读者也可以尝试用.NET Framework SDK 提供的命令行工具来编译代码。用那些命令行工具并不方便，但是它可以加深你对整个开发的理解，这样读者就不会只知道单击哪个按钮可以编译，单击哪个按钮可以运行。

1.3.2 公共语言运行库概述

我们可以将运行库看做一个在程序运行时管理代码的东西，它提供内存管理、线程管理和远程处理等核心服务，并且还强制实施严格的类型安全以及可提高安全性和可靠性的其他形式的代码准确性。

我们在.NET 编程中，编译生成的代码并不是最终的机器码，而是一种叫做 Microsoft 中间语言（MSIL）的代码。在运行时，由公共语言运行库提供的 JIT（Just In Time，实时编译器）把它“翻译”成特定的机器码，然后再执行。了解 Java 的朋友一定会想起 Java 的虚拟机，可以说，在一定程度上，这和 Java 的机制很像，但是在很大程度上又有其不同：

- 通过为不同语言提供编译成 Microsoft 中间语言（MSIL）的编译器，公共语言运行库可以支持多种语言（VC++.NET, VB.NET, C#.NET, ASP.NET 等）；
- 通过公共语言规范，提供了各种语言的互操作性。

下面简要介绍一下公共语言规范（Common Language Standard, CLS）。

如果我们写的对象要和其他语言的对象交互，那么其他语言的对象和我们的对象必须遵守一个同样的规范，这就是公共语言规范（CLS）。它是许多应用程序所需的一套基本编程语言功能。CLS 通过定义一组在多种语言中都可用的功能来增强和确保语言的互用性。CLS 还建立了 CLS 遵从性要求，帮助我们确定我们的托管代码是否符合 CLS，以及一个给定的工具对托管代码（使用 CLS 功能的代码）的支持程度。

如果我们的组件在对其他代码（包括派生类）公开的 API 中只使用了 CLS 功能，那么可以保证任何支持 CLS 的编程语言都可以访问该组件。遵守 CLS 规则并且仅使用 CLS 中所包含功能的组件叫做符合 CLS 的组件。

大多数.NET Framework 类库中的类型都符合 CLS。也就是说，我们可以在任何遵从 CLS 的编程语言（如 VB.NET，VC++.NET，C#.NET）中使用这些类型。但是，类库中的某些类型具有一个或多个不符合 CLS 的成员，这些成员支持 CLS 中没有的语言功能。

有了公共语言运行库，就可以很容易地设计出能够跨语言交互的组件和应用程序。也就是说，用不同语言编写的对象可以互相通信，并且它们的行为可以紧密集成。例如，可以定义一个类，然后使用不同的语言从原始类派生出另一个类或调用原始类的方法，还可以将一个类的实例传递到用不同语言编写的另一个类的方法中。这种跨语言集成之所以成为可能，是因为基于公共语言运行库的语言编译器和工具使用由公共语言运行库定义的通用类型系统，而且它们遵循公共语言运行库关于定义新类型以及创建、使用、保持和绑定到类型的规则。

1.4　Visual Studio 2005 开发环境介绍

C++作为一门编程语言，有众多可视化的开发工具，而 Microsoft Visual Studio 2005 作为一个功能相当强大的可视化应用程序开发环境，其中包括输入程序源代码的文本编辑器、设计用户界面的资源管理器以及检查程序错误的集成调试器等工具。

在编写应用程序之前，我们先简要地了解一下 Visual Studio 2005 IDE 的功能。

1.4.1　配置设置

每个开发者都有自己的开发风格，Visual Studio IDE 使你能够把各个 IDE 元素摆列成最适合自己的开发风格。

单击菜单栏中的“工具”→“选项”命令，打开如图 1.2 所示的对话框，其中提供了开发环境的设置类别。

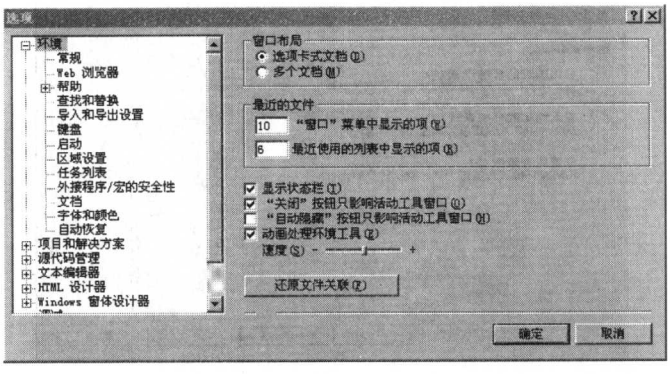

图 1.2　“选项”对话框

在 Visual Studio 2005 之前的版本，每次使用新的 IDE 都必须重新设置首选项，这让开发人员不胜其烦。Visual Studio 2005 提供了“导入和导出设置”选项，可以快速轻松地还原首选项设置。如图 1.3 所示，选中“选项”对话框中的“环境”选项中的“导入和导出设置”子项，“将我的设置自动保存到此文件”文本框中的配置文件在每次关闭 IDE 时都会进行更新。配置文件既可以是本地文件也可以是网络文件。如果是网络文件，在其中一台计算机上修改设置，则其他使用这一配置文件的计算机也会使用新的设置。

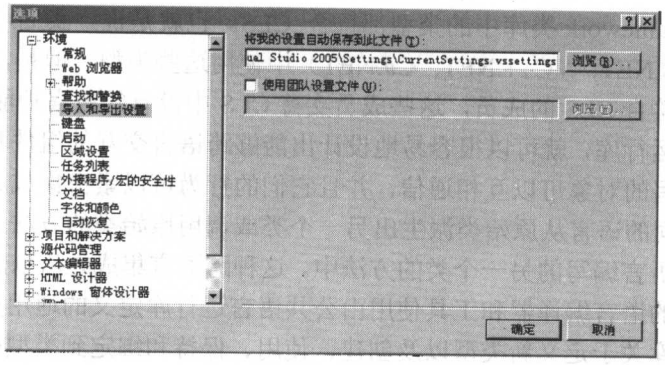

图 1.3 “选项”对话框

“使用团队设置文件”复选框可以指定一组开发人员共享的设置。只要把设置文件保存到网络共享的配置文件中，开发团队的人员就可以更新他们的 IDE 配置来使用团队的配置文件。如果修改了该网络共享的配置文件，则其他成员在下次启动 IDE 的时候将自动更新该设置。

1.4.2 导入和导出设置

在 Visual Studio 2005 中，选择菜单栏中的“工具”→“导入和导出设置”菜单项，打开“导入和导出设置向导”对话框，该对话框提供了导入或者导出特定的设置类别，或者将环境重设为一个默认设置集合，如图 1.4 所示。

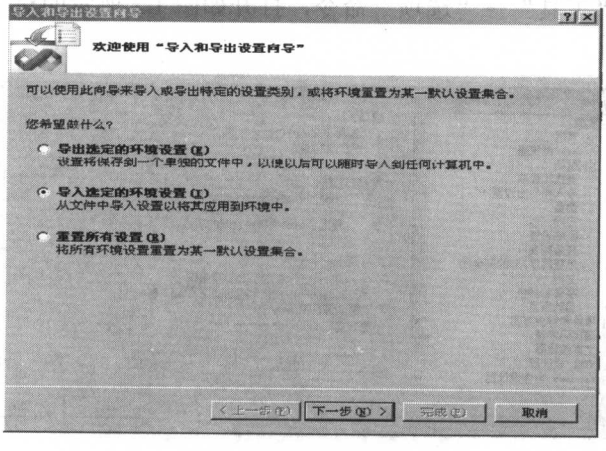

图 1.4 “导入和导出设置向导”对话框

1. 导出选定的环境设置

选择“导出选定的环境设置”单选项，并单击“下一步”按钮，出现如图 1.5 所示的对话框，从中选择要导出的设置，并单击“下一步”按钮出现如图 1.6 所示的对话框。重命名文件，并单击“完成”按钮，完成导出操作。

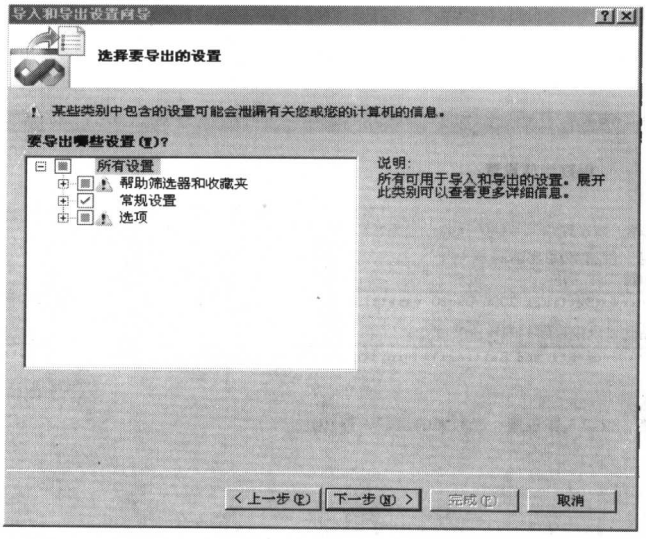

图 1.5　“选择要导出的设置”对话框

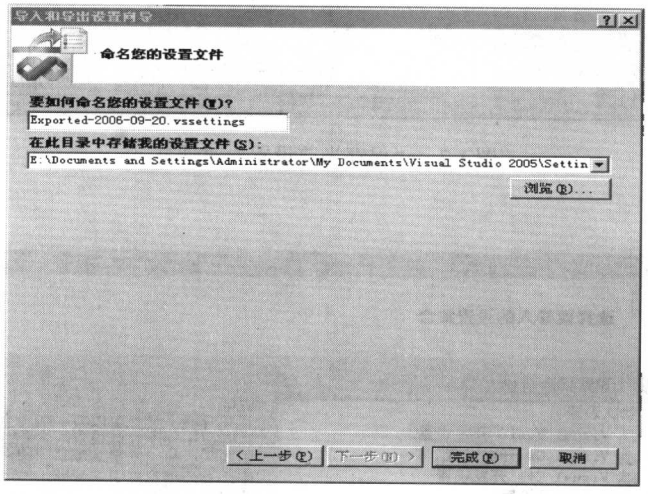

图 1.6　“命名设置文件”对话框

2. 导入选定的环境设置

选择“导入选定的环境设置”单选项，并单击“下一步”按钮，出现如图 1.7 所示的对话框，从中选择要导入的设置，并单击“下一步”按钮，出现如图 1.8 所示的对话框。选择要导入的设置集合，然后单击“下一步”按钮，出现如图 1.9 所示的对话框，选择要导入的设置，单击“完成”按钮，完成导入操作。

3. 重置所有设置

选择“重置所有设置”单选项，并单击“下一步”按钮，出现如图 1.10 所示的对话框。从中选择“是，保存我的当前设置”单选项，并输入文件名，然后单击“下一步”按钮，出现如图 1.11 所示的对话框，从中选择所要重置的设置集合，并单击“完成”按钮，完成重置所有设置的操作。