微软精品课程
系列

.NET
技术大系

“这本书的第一批读者是我教的大学一年级新生们，他们没有编程基础，就像璞玉未经雕琢。我每写一章，就给他们看并让他们告诉我哪里看不懂，然后我就修改——如此反复，直到他们完全看懂为止。所以我相信，编程的奥秘，每个人都能掌握。”

编程的奥秘

——.NET软件技术学习与实践

金旭亮 著

以 .NET 为载体，全面把握软件技术脉络
以实践为基础，直观体会程序设计本质

本书全部**示例源码**
网络热帖作者自传
本书配套**电子教案**

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

.NET
技术大系

编程的奥秘

——.NET软件技术学习与实践

金旭亮 著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

本书主要介绍在.NET 下开发运行于个人计算机上的应用程序所需的全部技术基础，汇集了从实践中总结出来的大量编程技巧与经验之谈，体现了作者对程序设计这一人类智力密集型活动的观点与看法，并力图为读者勾画出一个实用的软件开发学习全景，为读者进一步深入地自学相关的计算机专业课程（如“数据库原理”、“数据结构”、“操作系统”等）打下扎实的基础。

对于初学者，可以选择此书作为软件开发领域的入门书，一步到位，直接学习主流的面向对象软件技术；对于在校的大学生，学习本书则有助于了解计算机专业课理论如何应用在软件开发中，避免学习上的盲目性；对于已有一定编程经验的程序员，此书可以帮助其迅速地进入.NET 技术领域，结合其已有技能，开发出具有专业水准的应用软件。

本书所配光盘包含全书的全部示例源码；按章节组织，方便读者对每一章的深入学习；另外，还配有全书的电子教案及相关实例，方便将本书作为教材的老师授课用。

图书在版编目（CIP）数据

编程的奥秘——.NET 软件技术学习与实践 / 金旭亮著. —北京：电子工业出版社，2006. 1
（.NET 技术大系）
ISBN 7-121-01820-9

I. 编… II. 金… III. 计算机网络－程序设计 IV. TP393

中国版本图书馆 CIP 数据核字（2005）第 116436 号

责任编辑：孙学瑛
印　　刷：北京智力达印刷有限公司
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
经　　销：各地新华书店
开　　本：787×1 092　1/16　印张：38.75　字数：935 千字
印　　次：2006 年 1 月第 1 次印刷
印　　数：5000 册　定价：65.00 元（含光盘 1 张）

凡购买电子工业出版社的图书，如有缺损问题，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。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。

序 言

为响应教育部推出的“国家精品课程建设项目”精神，2004 年春季，微软亚洲研究院推出了“微软精品课程支持计划”。该计划旨在帮助提高 IT 相关课程教学质量，帮助高校基础性课程与企业的最新技术相结合。

该计划自推行以来，受到各高校老师的热烈响应。在微软亚洲研究院的支持下，高校一线教师不断努力创新，完善教学内容，提高教学质量，并将教学成果及时总结、共享。目前，多门经过系统整理、更新的课件资源共享在因特网上，其中的一部分还被吸纳到微软全球课程资源库中，供全球各高校教师交流、共享。在精品课件的基础上，经过几轮课程的教学工作，很多重点院校的教师，将课程进行了更系统的总结，整理成系列的教材及图书。这样，也就有了我们今天看到的这一系列内容上兼备基础性、科学性和前瞻性的丛书。

这套丛书计划覆盖微软相关核心技术，特别是微软最新.NET 及 WinCE 嵌入式系统技术。其中北京理工大学金旭亮老师所写的《编程的奥秘——.NET 软件技术学习与实践》是本套丛书的第一本，也是相当优秀的一本。金老师成功地在大学一年级开设了.NET 课程，并把自己多年的开发经验和感悟与最新的.NET 新技术融入教学，对学生“授之以渔”，使学生有了一个很好的起点。现在他又将教学成果整理成书，将使更多的开发人员和学生受益。

目前，丛书的其他分册正在编写过程中，在不久的将来会相继出版，在此我代表微软亚洲研究院感谢丛书各个编写小组教师的辛勤工作；感谢电子工业出版社的大力支持；希望本套丛书能够对相关的高校学生、专业工作者有所帮助，对中国软件产业起到很好的促进作用，发挥应有的效果。

今后，微软将一如既往地支持高校教学工作的开展，帮助高校及时了解微软最新技术，促进微软的核心技术与高校课程建设的紧密结合。而在此基础上，微软也将继续支持一线教师写出更多更好的书籍，为中国信息产业的发展尽自己的一份力量。

微软亚洲研究院 院长 沈向洋

自　序

这是一本很实用且独具特色的书。

这是一本讲技术，更讲学习方法的书。

这是一本从头至尾贯彻“授人与鱼，不如授人与渔”宗旨的书。

2003 年暑假，笔者在 CSDN“程序人生”论坛上发表的个人自传——《一个普通 IT 人的十年回顾》（已收入本书配套光盘），一石激起千层浪，被许多网站转载，我个人也收到了海内外近千封电子邮件。

我是一位在没有名师指导的情况下，几乎完全靠自己在黑暗中摸索，在自学之路上艰难地跋涉过来的软件开发者。我不敢自称为“职业程序员”，只敢自称为“软件开发爱好者”。我知道自己的技术水平有限，远远达不到“高手”与“专家”的水平，但我花了 10 年时间去学习计算机技术，其中酸甜苦辣，冷暖自知。我愿意把这期间的所思和所悟，与广大读者分享，并期望能帮助更多的初学者不走或少走我所走过的弯路，迅速地迈入软件开发技术的大门，最终成为远远超过本人水平的软件技术高手。

若能达此目的，则我在这本书中所花的心血也就值得了。

我为什么写这本书

我是在大学毕业后才开始半路出家学电脑的，走了一条艰难曲折的自学之路。在写作本书的时候，刚好满 10 年。10 年以来，我几乎就没离开过计算机，而编程更是从未间断。写过的代码到底有多少恐怕无法统计了（加起来肯定有几十万行）。出于对软件技术的强烈兴趣，我从对计算机一无所知起步，到后来考上了计算机专业的研究生，毕业之后又走上高校计算机教学岗位，感触颇多。

在有了 10 年自学计算机技术的经历和近几年在高校讲授程序设计系列课程的经验之后，我一直在思索以下几个问题：

- 软件到底是怎样开发出来的？
- 编程是难还是易？有没有能让编程功力“暴长 10 年”的“武功秘笈”？
- 一个有志于成为优秀软件工程师的人应该怎样学习计算机技术？
- 用什么方式才能让许多学生很快地学会特定的计算机技术，而不用重复我和其他人所走过的弯路？

思索的结果就是大家所看到的这本书。

在我自己亲身的学习与开发实践中，逐渐形成了这么一个观点：

编程其实并不难！

要编程并不需要到大学计算机系里去学很多艰深的理论，经过适当的指导与训练，一个高中

生就可以参加到软件项目团队中，并写出具有相当质量水准的程序！

为什么会有那么多的人把编程视为很高深的东西？

为什么我国那么多的软件企业深感合格的程序员是如此难找？

与此同时，又有与之完全对立的情况：每年有大批计算机及相关专业毕业生四处求职，却处处碰壁，找工作成了一件很不轻松的事情……

一边是真正合格的程序员人数很少，软件企业都在喊急缺人才，而另一边是大量的学生毕业找不着理想的工作。这样一个怪圈为何会形成？

我想，造成这个怪圈有两个重要的原因。

一是学生学习方法与态度的问题。由于对软件开发这一工作本身缺乏正确认识，许多学生形成了一种浮躁的功利型学习方法，表现为盲目地跟风学习各种当前流行的新技术，但大都只是三个月热度，由于欠缺毅力和不具备扎实的理论与实践基础，对这些新技术无法深入把握，浅尝辄止，“坐而论道”，讨论时可以“滔滔不绝”，一到动手则“原形毕露”。许多人没有想过：现在流行的“新”技术，难道就会一直“流行”下去？曾经风光一时的 FoxPro 和 PowerBuilder，国内曾有许多人（包括本人在内）投入大量的时间去学习与掌握，可没过几年，还有几个项目是用这两种技术开发的呢？所以，如果要“追”新技术，也只能去学预计两到三年后社会急需的技术，这就需要学习者有超前的眼光。但软件技术进步实在太快，很少有人能够准确预测出几年之后会流行什么。所以，与其“临渊羡鱼”，不如“退而结网”，选择一项自己喜欢的领域，从理论到技术，扎扎实实地用两到三年时间去学好它，在学习过程中培养出“捕鱼”的本事。有了这种自我学习的基本能力与素质，还担心出海捕不到鱼吗？

有一些学生投入相当多的时间与金钱去参加种种有着各种动听承诺的电脑培训班，指望别人能够帮助自己成为技术高手，却不知真正的高手大都是“自学成才”！

更有太多的尤其是非计算机专业的学生，盲目地去考各种认证，不管是国内的还是国外的考试，只要有证，就有大批的人去参加。他们把应用计算机技术的能力简单地等同于一纸证书。当前的计算机等级考试就大有成为高校中第二个“英语四六级”的可能性。**事实上，你向别人展示自己写的一个软件作品，比给别人看一堆各式各样的证书更能说明你的能力！**

造成怪圈的第二个重要原因就是计算机教育问题。目前高校中的计算机教育并不能大批量地培养出合格的软件开发者，学生在四年本科期间计算机理论学了不少，对培养编程能力却重视不够，缺乏实践的结果是理论也掌握不好，学生的计算机水平被煮成了“夹生饭”。

除了在校的大学生，社会上还有大批的软件开发爱好者，他们由于没有机会系统地学习软件开发技术，就采用自学和自我实践的方式，结合自己的工作来应用学到的计算机技术。虽然他们没有在校生的优越条件，但其中优秀人物的真实开发能力远胜于正规高校计算机专业“科班”出身的本科生甚至是硕士生。这种例子已经很多了。

“职业选手”不如“业余选手”！

我自己是走“野路子”出来的，走的是以实践为主导的自学之路，从亲身实践中深切感受到计算机科学本质上是一门实践性非常强的科学，不管是在校学生还是社会上的软件开发爱好者，我的建议是：

实践就是最好的老师！

另一个需要强调的是：

编程是一种技能，就同学习外语一样，绝不可能“速成”！

我从2002年开始接触.NET技术，并一直研究使用至今，前前后后看了近百本国内外的.NET编程书籍，并写了几万行的.NET代码。在对.NET有了一定的了解之后，我认为.NET技术是初学者学习软件技术非常好的切入点，学习者可以一步到位，直接学习目前主流的面向对象软件开发技术，而不用按部就班，按传统的先结构化编程再面向对象编程的顺序去学习。

接着我思考的问题是：这本书该怎么写？

著名物理学家爱因斯坦曾经表达过这样的思想：

学习时只需要掌握那些能让思维导向深入的东西，而将其他无关的知识全部抛弃。

软件技术本身有一个庞大而复杂的体系结构，不分巨细地样样都想掌握，那么只能是疲于奔命而终无所得。

为此，基于我自己的理解，在书中我将认为是最重要和最基础的东西凸显出来，这些东西是必须掌握的，一旦理解并掌握它们之后，读者就掌握了“捕鱼”的方法，自己就可以进一步探索某一领域内的其他技术点了。因此在书中不需对每个技术领域的每个技术点都“面面俱到”。

编程是有规律的，读者能不能迅速成为一名具备相当水平的程序员，关键在于对编程规律与必备知识基础的把握程度。

读者在阅读本书时，最重要的是要掌握原理。在开发过程中所进行的各项具体活动与工作步骤，其背后都是有道理的，理解清楚了这些道理，在整个软件开发过程中该做什么，不该做什么，先做什么，后做什么，也就心中有数了。

编了这么多年的程序，我意识到其实学习软件开发技术的过程就是一个学会做事方法的过程。例如，在开发软件时，要“分而治之”、“步步为营”、“先局部后整体”、“合理安排开发次序”、“养成良好的编程习惯”等，这不仅仅是程序设计的技巧，也是做好编程之外许多其他工作所需要的。

在学习与掌握软件技术这一过程中，可以锻炼一个人的许多基本素质，如思维周密、意志坚强、学习能力、创新精神等。如果一个人能在竞争空前激烈的IT业凭借着高素质站稳脚跟，那么，可以很有把握地说，如果他转向其他行业，同样是一个优秀的人才，也会闯出自己的一番天地！

在本书中，结合我个人的开发经验，精心安排了学习次序，循序渐进地引导读者迈入软件开发的大门。

在许多计算机技术书籍中，出现了大量的技术术语，对于不熟悉这些术语的读者而言，这无疑加大了阅读的难度。对于这些计算机术语的表述，如果强调其在理论上的科学性与严谨性，则往往不易于理解，这在很大程度上限制了这些知识的传播、普及和应用。因此，本书对这些术语大都采用了一种通俗易懂的方式来表达，甚至使用了不少比喻手法来阐明其内涵。这种表达方式虽易于理解，但不免在科学性和严谨性上有所不足，读者如果需要科学而精确的定义，请自行查阅相关的科技文献。强调一下，本书是一本志在普及软件开发技术的专业书籍，与传统的工程技术专著和计算机科学理论教材不同，阅读时不要像中学学习数学一样，每个字都不放过，而应将重点放在形成对软件开发过程的直观理解和感性认识上。

对于一个具体的软件开发者而言，只有被理解了的知识才是真正有用的知识，无法理解的知识是毫无用处的。笔者写书的主要目的是让读者能迅速地汲取书中所提供的知识，尽量做到**“用大白话讲述复杂的技术”**，因此在文字上追求深入浅出，但是否真的做到了这一点，还得由读者来做出评价。

知识的获取与掌握是有其客观规律的，一般而言，具体的知识易于理解，而抽象的知识则难于把握，在学习过程中，应遵循“具体→抽象→具体→……”这样一个无限循环的过程。

如果某人对一门学科所知不多，却指望他能通过理论学习迅速地把握这门学科，这就违反了人类认识世界的客观规律。可惜的是，目前中国教育体制中普遍采用的让学生重点学习抽象理论而忽视具体实践的做法，造成了大批的学生既没学好理论，也没打好必要的实践基础。

我认为，只有建立在实践基础之上的理论学习才是真正有效的学习。因此在整个计算机技术的学习过程中，应该把开发实践作为一条主线，由它串起所有的计算机理论。如果把计算机技术中的各个子领域看做是一个个珍珠，开发实践就是那根把所有珍珠串成项链的丝线。没有这根线，学习各门计算机课程得到的不过是一颗颗零散的珠子，价值有限。

出于这些考虑，笔者在介绍许多编程技巧的同时，也同步介绍了相关的高等数学、数据结构、操作系统、数据库理论、软件工程等计算机科学理论内容，其目的就是希望读者能了解到理论是如何应用在开发实践中的，从而有助于读者日后针对性更强地去学习这些计算机理论课程，最终成为一名优秀的软件工程师。

本书虽然介绍了大量的编程技巧，但绝不是编程技巧的简单汇编，其中涉及了计算机科学中相当广泛的知识。如果读者能够从本书中体会到计算机技术的脉络，并培养出一定的开发能力，那么我会感到非常高兴与欣慰。

学习此书，要求读者对 Windows 操作系统比较熟悉，对软件开发有着浓厚的兴趣，并不要求读者系统学习过计算机专业课程，或者有 C、Pascal 等丰富的程序开发经验。

怎样学习软件开发技术

1．区分计算机科学与计算机技术

“科学技术”四个字往往连在一起说，但事实上“科学”与“技术”是有区别的。

计算机科学主要研究理论，而计算机技术则是这些理论在实践中的应用。打个比方：编译理论是科学，而具体的编译器，如 VBC（微软公司开发的 VB.NET 程序编译器）就是技术了；操作系统原理是理论，而 Windows 就是理论应用于实践的产品，因而归属于技术范畴。

讲到这里，读者可能明白了，计算机科学家就是指那些从事计算机理论研究的人。目前，我国高校中计算机系开设的大部分课程都属于计算机科学范畴，比如“数据库原理”、“操作系统”、“计算机图形学”、“人工智能”、“算法理论”、“数据结构”等；而另一些课程则属于技术范畴，比如“Visual Basic 程序设计”、“SQL Server 数据库开发”等。还有一些课程，比如“面向对象程序设计”、“软件工程实践”等往往同时涉及理论和技术，因而是“混血儿”。

“科学”与“技术”的区别，自然地将从事计算机工作的人分成了两大块：计算机科学研究者与计算机技术工程师。前者研究的是理论，是计算机科学，人数较少，他们从事的是往计算机科学大厦中添砖加瓦的工作，是新知识的发现者。一般而言，只有非常优秀的人才能进行理论研究工作。后者研究的是技术，属于工程范畴，他们关注的是如何把一个好的软件、一个好的系统实实在在地做出来，大部分人属于工程范畴。

计算机技术总体上又分为软件与硬件两大块：硬件工程师研究的是芯片、集成电路、板卡等“实实在在”、可以“摸得着”的东西；而软件工程师则编写程序，驱动各种各样的硬件完成工作。

在计算机技术中，有一个并不严格但众所周知的“软硬定律”：

任何一项由软件完成的功能，也可以用硬件实现，反之亦然。

比如 Java 虚拟机（Virtual Machine），它原先属于软件范畴，但如果将整个虚拟机烧录到硬件芯片中，然后在它上面运行 Java 程序，那你说现在这个虚拟机是“硬件”还是“软件”？这个界

限已经模糊了。

我们回到本书所介绍的.NET，.NET是属于软件技术范畴的。换句话说，如果你想成为一名软件工程师，就可以看这本书。

2. .NET 与 Java

Java 是 20 世纪 90 年代诞生的一种面向对象的语言，它吸取了 C++的许多长处，其最引人注目的特性是“跨平台”。由于它很好地满足了网络时代信息系统对计算机高级语言的要求，因而在实践中得到了广泛应用。

Java 经过近 10 年的发展，目前已在企业信息系统开发领域占据了主流地位，形成了一整套完整的技术体系，取得了巨大的成功。

.NET 是微软推出的一种新的软件运行平台，它包括一整套应用范围广泛的技术，旨在迎接 Java 所带来的挑战。.NET 给软件开发带来了很多新的技术，尤其是它的“混合语言”开发，是第一个成熟的支持多种语言混合开发方式的软件运行平台，在下一代 Windows——LongHorn（预计 2006 年推出）的核心中将会看到这个新技术的身影。

如果读者学过 Java，就会发现.NET 技术与 Java 技术在许多方面实在是太像了。尤其是 C#（.NET 下的一种编程语言，念成“C Sharp”，不要念成“C 井”☺），如果给你一段代码，有的时候还真难分辨出是用 Java 还是 C#写的。

.NET 与 Java 技术的这种相似为我们的学习提供了便利，举个实例，如果你掌握了 C#，那么掌握 Java 这个语言几乎不费什么力气，需要花费时间的只是熟悉彼此的类库。.NET 的标准类库是.NET Framework，Java 的标准类库是 J2SE（注：Java 类库还包括用于手机等移动设备开发的 J2ME 和用于企业级信息系统开发的 J2EE）。掌握这两种技术的基础都是面向对象理论。所以，一名有着扎实的面向对象理论基础的人在学习.NET 和 Java 技术时就可以触类旁通，举一反三。

.NET 学习全景图

.NET 有一个庞大的技术体系，初学者如何决定自己的学习步骤呢？就我自己的经验，谈谈对这个问题的看法，如图 1 所示。

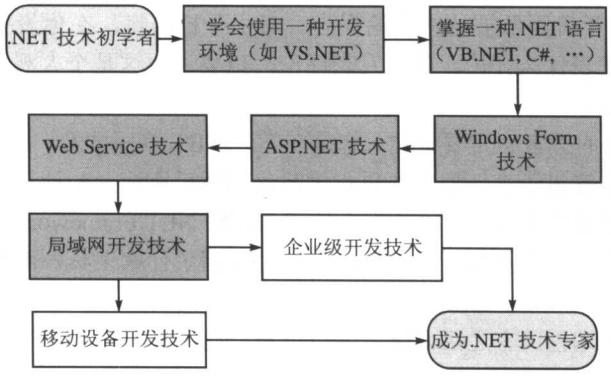

图 1 .NET 学习全景图

作为一个下定决心要学习.NET 的初学者，第一步是要学会使用一种开发环境，如 VS.NET，之所以把掌握这一开发工具放在第一位，是因为在后面的学习过程中，需要使用 VS .NET 进行不断的编程实践，而只有通过实践，才能最终掌握.NET 技术。

第二步是掌握一种.NET 语言。由于.NET 支持多种语言，所以读者可以根据自己的喜好选择

一种计算机语言，但请注意以下选择语言的标准：

（1）必须是全面向对象的语言，只有面向对象的语言才能充分发挥.NET 的威力；

（2）该语言必须拥有较多的开发者和技术资源，这样你就可以很容易地找到志同道合的人共同学习，有问题也容易找到高手解答。

微软官方提供的语言有以下几种：Managed C++、C#、Visual Basic .NET 和 J#。

其中 Managed C++是对标准 C++的一种扩充，用的人很少（目前 Managed C++将演变为 CLI，但其前景仍不十分明朗）；J#则是微软为吸引 Java 程序员转到.NET 平台而采取的一种策略。如果读者不属于原来使用 C++和 Java 开发的人群，并希望能继续使用原有的语言写.NET 程序这种情况，就不要学习 Managed C++和 J#。

C#和 Visual Basic.NET 是目前.NET 下开发的主流语言。其中，C#是全新设计的一种语言，其语法与 C++和 Java 都非常类似。Visual Basic.NET 保留了原来 Visual Basic 的大部分语法特征和简洁方便的特性，同时加上了对面向对象特性的全面支持。两者几乎是一样强大的，初学者选任何一种均可。

目前支持.NET 的语言已有几十种，而且还在不断地增加中。

本书选用的编程语言是 Visual Basic .NET。之所以选用 Visual Basic .NET，有两个原因：一个原因是本书作者从 Visual Basic 3.0 开始就使用 Visual Basic 编程，有丰富的 Visual Basic 开发经验；另一个原因是.NET 支持混合语言开发，用 Visual Basic .NET 开发出来的组件，可以方便地组合到 C# 程序中，反之亦然。由于 Visual Basic .NET 与 C#都使用同一个类库——.NET Framework，因此把 Visual Basic .NET 的程序代码转成 C#代码也并不困难。

本书正文中的全部实例均以 Visual Basic.NET 语言开发，但并不要求读者以前学过 Visual Basic。当然，如果使用以前版本的 Visual Basic（如 Visual Basic 6）开发过程序，那么，使用 Visual Basic .NET 会感到比较亲切。网上有许多人对 Visual Basic 有成见，许多初学者为了学.NET 编程而一起跟风去学 C#，其实 Visual Basic 是一种优秀的计算机编程语言，尤其是最新的 Visual Basic .NET，更是比以前的 Visual Basic 强大和灵活。笔者使用过许多其他的语言（如 C++、Pascal、C#和 Java）开发过程序，但 Visual Basic 许多独特的优点所带来的高效率开发仍是让笔者情有独钟。

在此，我只想说一句：关于语言优劣的讨论意义并不大，网上许多人所谓的高论，其实都是“盲人摸象”罢了，初学者不要被这些言论误导。对于一种语言，在你没有使用它写过一定量的程序之前，不要对其发表什么评论。就算发表评论，也要说明这是个人观点。这是一名软件工程师对技术应采取的态度。

所以，在学习本书时读者一定要记住，您是在学习.NET 编程技术，在学习如何开发软件，而不是在仅仅学习一门计算机语言。您需要重点学习代码背后所涉及的计算机基础理论知识，把握代码中所蕴涵的软件开发思维方法，以及学会使用并理解.NET Framework 本身。掌握编程语言不是重点，语言只不过是一个工具罢了。重要的不是您掌握了什么工具，而是您用这个工具做出了什么！

掌握了开发工具和编程语言之后，读者就可以自己动手编一些小程序了。接下来的任务是学习具体的编程技术。

.NET 上运行的大部分程序都是基于网络的（比如用 ASP.NET 开发的网站），但对于初学者而言，不适合一下子就投入到网络程序的开发中，而应从易到难，先把基础打好。

因此，在第三步中学习 Windows Form 技术是个好的选择。所谓 Windows Form 程序，就是传统的运行于个人电脑上的单机应用程序，如大家常用的金山词霸。通过对 Windows Form 技术的学习，读者可以掌握最重要、最基础的 Windows 软件开发技术，并对面向对象理论有一定的体会。

有了这些基础，深入学习网络编程就是水到渠成的事。本书的主要内容就是向读者介绍 Windows Form 技术。

第四步则是学习.NET 网络技术，具体可以按照以下顺序进行学习。

（1）ASP.NET：主要掌握如何设计 ASP.NET 网页，搭建运行于.NET Framework 之上的 Web 网站。

（2）Web Services：学习如何把网站的各种功能以 Web Services 的形式提供给外界，或者集成外部的 Web Services 以实现功能强大的信息系统。Web Services 是一种潜力无穷的新技术，将对下一代的因特网产生深远的影响。

（3）.NET Remoting 及相关的局域网技术：不像 ASP.NET 和 Web Service 主要用于因特网，.NET Remoting 及相关的技术主要用于开发基于局域网的应用程序。目前大部分公司和企业都建有自己的局域网，开发基于局域网的信息系统具有很大的需求。

当完成了以上四步的学习之后，您就基本上了解了.NET 中最重要的技术，下一步有两个选择：

（1）一个方向是学习移动设备开发技术，比如为手机、SmartPhone 等智能移动设备开发程序，设计嵌入式系统等，.NET Compact Framework 就是专为移动设备应用程序开发而准备的，使用.NET Compact Framework 开发移动设备程序与使用.NET Framework 开发普通程序是类似的，已有的.NET Framework 开发经验仍然适用。

（2）另一个方向是进一步深入地学习企业级项目开发技术，成为一名系统架构设计师。这需要更进一步地学习相关理论和技术，比如设计模式、软件工程等，还有如 J2EE 等的.NET 竞争对手的技术。

如果读者能沿着我的路线图走到这里，那就恭喜你，你已成为了一名.NET 技术专家，更宽广的道路将展现在你的面前。

本书写作的主要目的之一是帮助想学习.NET 技术的读者顺利地走完前三步，为以后的路打基础。

理解本书内容的安排次序

本书主要介绍了开发一个功能完备的.NET Windows Form 应用程序所需要掌握的全部技术基础。没有任何.NET 开发经验的人也完全可以通过此书迈入.NET 软件开发的大门。

在内容安排上，与通常的编程技术书籍及教材有所不同。我把全部内容分为循序渐进的四大部分，引导读者由浅入深地步入软件开发的大门。

在学习编程技术方面，传统的方法是让学生先学 C 语言。C 语言是一种结构化的编程语言，在软件史上有着独特的地位，它与计算机技术的各个领域密切相关，因此先学习 C 语言有助于学生在初期就接受严格的软件开发职业训练，让学生从只会以“人”的角度思维逐步过渡到以“计算机”的角度思维，并开始形成对“计算机是什么”这个问题的直观理解。如果 C 语言学得好，那么他日后学习 C++和 Java 等面向对象的程序设计语言，以及计算机原理、数据结构等专业课程就有了扎实的基础。

从“人”的思维转化到“计算机”的思维，是衡量一个人是否迈进了软件开发大门的重要标准。有多少学生在计算机专业学了四年本科课程之后，仍然不能自如地在“人”与“计算机”的这两种思维方式中转换！

不按照“计算机”的方式进行思维，怎么可能开发出控制计算机工作的程序？

因此，在现有的高校计算机教育体制中将 C 语言作为计算机专业学生学习的第一门高级语言

是有道理的。

但现在的问题是：从“人”的思维转换到“计算机”的思维这个过程是比较艰难的。有许许多多的学生因为不能完成这个转换而无法在毕业时成为合格的软件工程师。

C 语言是个相当复杂而精妙的语言，要掌握它实非短短一个学期的学习所能完成。在目前的中国高校中，普遍采用 Turbo C 进行 C 语言的编程训练，但 Turbo C 是基于 DOS 的，这个环境与目前主流的图形界面操作系统环境有本质的差别。学生会发现，学 C 语言非常辛苦，可学了好久却连一个真正实用的程序也编不出来。这种打击是巨大的，它很难让大多数学生体会到编程的乐趣，反而只体会到编程之“难”！学生一入学就吃了这个闷棍，导致对计算机兴趣大减，使后面更复杂更抽象的专业课学习变得难以忍受。

我个人认为，通过 C 语言来对一个在进入大学以前对编程知之甚少的学生（这种情形目前来说占有相当的比例）进行这种思维转换的训练，成功的比率并不高。

一个人只有对一件事情感兴趣，才可能主动地去学习相关的知识，并最终把这件事情做好。因此，对于初学者而言，引发他对于软件开发的强烈兴趣才是最重要的，不要一开始就用很复杂的东西去打击他的积极性。

正是基于这种考虑，本书的第 1 篇是初识阶段，让读者在第 1 章就对.NET 是什么有一个感性的认识，并且马上就可以试着编出第一个程序。紧接着在第 2 章就开始介绍可视化编程的内容，充分利用现代软件集成开发环境开发软件的高效性，让读者看到居然可以通过“绘图”的方式，迅速地“画”出一个真实的程序来，一切竟是如此的简单！紧跟着就在 2.2 节中介绍 VB.NET 的语法，让读者开始学会如何用代码来“控制一切”，自己来当软件的“主宰”。第 2 章的最后一节则以一个文本编辑器 MyEditor（参见图 2）的开发全过程作为实例，让读者看到一个“真实的”和“有用的”程序是怎样做出来的。MyEditor 包含数百行代码，写得非常规范，并加有大量的中文注释，读者通过阅读这个实例程序，可以很快地学会许多编程技巧，并激发起开发自己的软件的强烈兴趣。

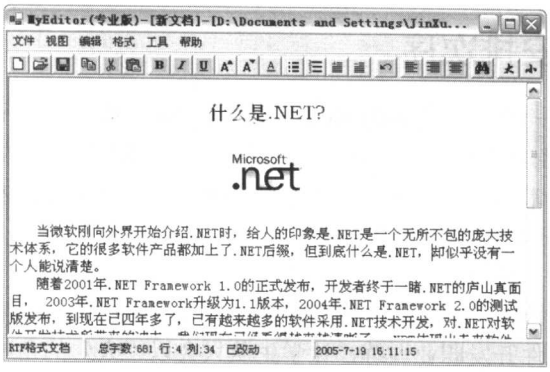

图 2　功能强大的文本编辑器——MyEditor 专业版

由第 1 章和第 2 章构成本书第 1 篇，其主要目的就是引发读者对.NET 软件技术的兴趣，并对这一先进的软件运行平台有一个直观的认识。

本书的第 2 篇取名为“入门”，在第 1 篇的基础上开始全面深化编程的基本技能。第 2 篇的开篇就是对 VB.NET 这一面向对象语言的全面介绍，占据了第 3 章的整个篇幅。读者如果看过本书目录，可能会奇怪，第 2 章的 2.2 节不是已经介绍过 VB.NET 语法了吗？怎么又重复了？

在第 2 章中，介绍的仅仅只是 VB.NET 中用于结构化编程部分的内容，而.NET 是一个全面向对象的软件运行平台，只有面向对象的语言才能充分发挥它的威力。VB.NET 的面向对象语言

特性非常丰富，即使用一章完整的篇幅，仍然不可能描述出它的全貌。为此，本章只是抽取出最重要的一些特性进行介绍，比如类、封装、继承、多态、委托等，但并不以介绍语法为重点，重点介绍的是这些特性如何用在程序代码中。我为每一个语法特性都设计了短小精悍的实例，并在正文中提供了大量让读者进行实地修改程序的实验指导，让读者亲身体会面向对象软件编程的特点。

如果读者以前没有学过一种面向对象的语言，请务必仔细地阅读第 3 章，并在 VS .NET 中把书中所有的例子全部运行一次。

强调：虽然配套光盘中有本章全部示例源代码，但我强烈要求以前没有学过面向对象编程语言的读者动手一行行地敲入代码。

按照许多技术书籍的惯例，往往是在全书的开头一次性地把所有语法讲完，而本书则将对 VB.NET 语法的介绍分为结构化与面向对象两部分，这是经过仔细考虑后做出的独特安排。

有了前 3 章的铺垫，第 4 章就可以讨论 Windows 编程中更深入的知识了。这一章集中介绍传统的运行于个人电脑上的 Windows Form 应用程序开发技术，包括使用文件、灵活使用控件、数据验证、键盘与鼠标操作等，内容非常丰富，全部都是开发一个真正的功能完备的应用程序所必须掌握的技术基础。

第 4 章以一个克隆 Windows 资源管理器的应用程序（参见图 3）的分析作为结束，让读者初步体会到如何把学到的零散的程序技巧组合在一起，最终完成一个有着相当复杂程度的软件。一旦掌握了这些技术，读者就基本上具备了开发 Windows Form 应用程序的能力，就可以投入到无限精彩的软件开发世界中去了……

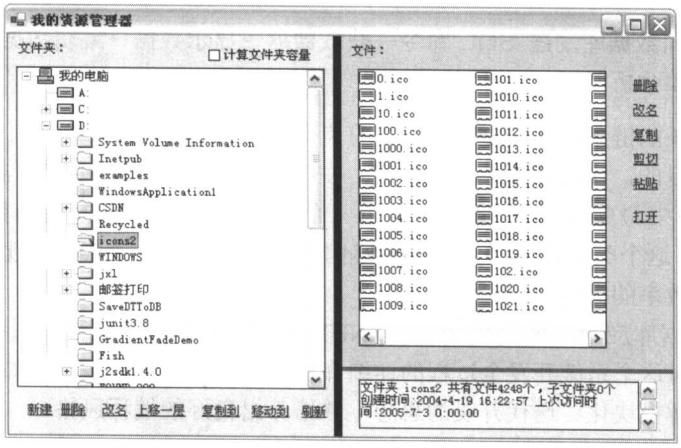

图 3　克隆 Windows 资源管理器——MyComputerExplorer

第 3 篇开始介绍.NET 中各个专业技术领域的内容。这是在前两部分学习的基础之上的深入，正因为如此，这一部分被命名为“深入”。

.NET 有着一个非常庞大的技术体系，任何一个人想在短期内面面俱到都是不可能的。为此，我选取了.NET Framework 中最成熟、最精彩，也最实用的三个技术领域：数据库、XML 和 GDI+，带领读者一览.NET 的风采。

首先是数据库技术。目前，中国各个行业都在进行信息化建设，因此对 MIS（Management Information System，管理信息系统）有着巨大的需求，这是一个充满了机会的巨大市场。而几乎所有的信息系统都是围绕着数据库建立起来的，数据库技术无疑在当前的信息系统中处于中心的地位。与以前的技术相比，微软公司在数据库应用程序开发技术上有了长足的进步，其核心技术

ADO.NET 给软件开发所带来的高效率、强功能和灵活性，是.NET 中一道亮丽的风景线。而掌握好数据库技术，不仅具有很大的实际意义，而且对读者日后进一步学习因特网编程技术至关重要。

正是由于数据库技术的特殊重要性，本书用了整整四章的内容来介绍。

“盲人摸象”是读者所熟知的一个成语，不幸的是，面对着庞大的软件技术体系，我们也常常无可奈何地处于“盲人摸象”的境地。笔者经过长时间的摸索与实践，终于在.NET 所提供的庞大数据库技术体系中理出了一条清晰的线索，帮助读者避免重蹈“盲人摸象”的困境。第 5 章“.NET 数据库编程概览”就是出于这个目的而组织的，通过一个典型的实例，介绍如何通过.NET 的 ADO.NET 对象模型访问数据库，让读者快速地了解使用.NET 开发数据库应用程序的概貌，避免在今后的学习过程中陷入“只在此山中，云深不知处”的困境。

凡是有过真正信息系统开发经验的软件工程师，无不对 SQL（Structured Query Language，结构化查询语言）喜爱有加。SQL 语言是专门用于操作关系数据库中数据的计算机语言，语法简洁而功能强大，在现在的软件系统中应用极为广泛。一条精心设计的 SQL 命令，往往可以代替几十行 VB.NET 或 C#代码！然而不幸的是，SQL 命令同样是一个庞大的技术领域，其中的编程技巧更是不胜枚举。第 6 章将从实用的角度出发，从中挑选出最基础、最实用的内容，让读者学会使用 SQL 命令操作数据的基本技巧，为读者自己学习与探索更多的 SQL 技术打下良好的基础。

第 7 章“编程访问数据库”集中介绍.NET 新的数据库存取引擎——ADO.NET，这是本书第 3 篇的重点章节。

ADO.NET 有一个庞大而复杂的体系结构，如何帮助读者快速地把握它？笔者思索了很久，最终决定以数据处理流程为中心，按照以下脉络串起整章内容：

连接数据库→向数据库发送 SQL 命令→获取数据→显示数据→编辑数据→保存数据→查找与过滤数据→统计和分析。

看了上面这长长的链条，读者是不是心中有点数了呢？

学习本身不是目的，实践才是最终的归宿。在第 8 章“数据库编程技巧与实例分析”中，我把数据库应用程序中最实用的功能封装成了一个功能强大、使用方便的数据库存取类——OLEDBAccessObj。这个类将在本书后面许多实例程序中一再复用，从而让读者初步感受到代码复用所带来的开发效率的提高。

第二个大的数据库应用实例是使用 ADO.NET 开发一个客户信息管理系统 ClientInfoSystem（参见图 4）。通过对这个系统开发全过程的详细介绍，使读者可以初步学会如何开发一个功能完整的软件系统，理解模块化、迭代开发的思想，并培养出良好的编程风格。

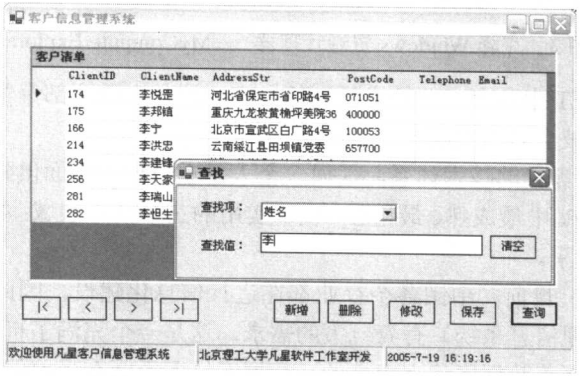

图 4　小巧灵活的 MIS——客户信息管理系统

学完了本书第 3 篇的前四章，读者一定会对 ADO.NET 功能如此强大而灵活深有体会。其实，在 ADO.NET 技术的内部，是采用 XML 来表达数据的。要想真正地理解 ADO.NET，还必须继续学习 XML 技术。

事实上，学习 XML 技术更重要的理由是：

XML 将是下一代因特网技术的基石！

目前非常红火的 Web Services（Web 服务）技术，就是建立在 XML 的基础之上的。而.NET 从一诞生开始，就具备了强大的 XML 支持能力，因此，在第 9 章“使用 XML 处理数据”中先介绍 XML 的基础知识，接着介绍.NET Framework 中用于处理 XML 数据的两种主要方式：DOM 和流机制。

本章实例包括可以直观查看 XML 文档 DOM 结构的 DOM 阅读器，以数据流方式在 XML 文件中查找数据的自定义类 XMLReaderObj，以及一个实用“公文包”小程序（这是一个可以脱机使用数据库的小程序，展示了开发移动办公软件的基本技术）。

XML 是一个正在发展中的技术领域，其应用日益广泛，读者日后应在本书所介绍知识的基础上，自行学习更多的 XML 技术和理论。

现在到了本书第 3 篇的最后一章——第 10 章“计算机绘图原理及其应用”。

这是内容异常丰富的一章，相信也是读者认为最有趣的一章。

计算机绘图无疑是最引人入胜、令人惊叹的部分之一。然而很遗憾，本书不是专门介绍计算机绘图技术的，因而只能从这浩瀚的科学与艺术结合得最紧密的知识海洋中拾起几个美丽的小贝壳，向读者介绍知识之海的广阔无边，并期望能小中见大，让读者初窥计算机图形学这一领域的奥秘……

本章分为三个部分。

第一部分介绍计算机绘图原理：首先介绍了使用 GDI+在处理图形方面的基础知识与编程技巧，并在此基础上引入了绘制数学函数图形的基本方法（参见图 5）。

利用递归这一特殊的编程技术，结合坐标变换、分形算法等数学知识，可以创造出精美的图案（参见图 6）。

第 10 章中还介绍了处理图像的基本原理与方法，掌握了这些基本的知识与技能，读者不仅可以开发出自己的矢量绘图软件，而且可以更进一步地去学习《计算机图形学》等专著。在这部分中，读者还可以直观地看到数学知识与计算机绘图技术是如何结合在一起的。

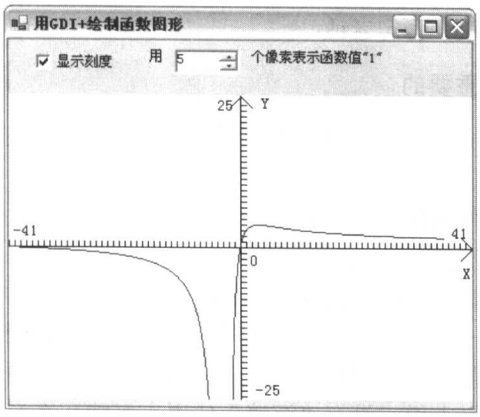

图 5　绘制数学函数图形

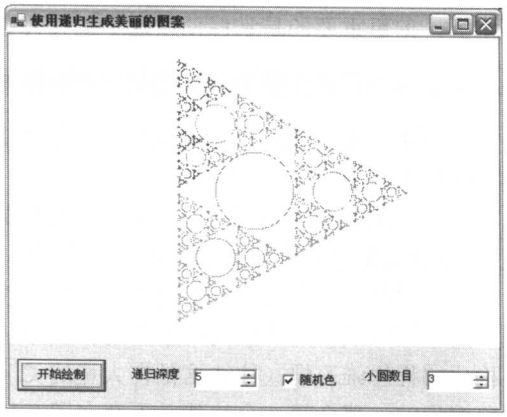

图 6　递归分形

第 10 章的第二部分介绍 GDIClock，此程序是微软公司在 Visual Basic .NET How-to 范例库中提供的，其开发者是 Ken Getz（参见图 7）。

阅读并理解别人写的代码，是软件工程师最重要的基本技能之一。摔跤、柔道等运动员都知道：要想胜人，先学挨摔。同样地，要想成为优秀的软件工程师，就要先学会如何阅读别人的代码，“择其善者而从之，其不善者而改之”，自然就能不断进步。

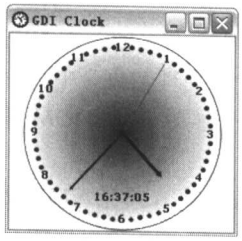

图 7　精美的小时钟

GDI Clock 程序使用 GDI+技术绘制时钟，并在编程中应用了许多技巧，是一个非常好的学习范例。笔者对整个软件进行了汉化，并在源代码中加入了大量的中文注释，期望读者仔细剖析这一优秀程序实例。

打印是应用程序中非常重要的功能，令初学者很困惑的是，为什么“打印”与“绘图”居然是“一家人”？ 第 10 章的第三部分将为您解开谜团。10.3 节首先介绍打印的基本原理，接着用循序渐进的几个小实例全面介绍在.NET Framework 下开发打印程序的基本技巧，帮助读者理出一条清晰的脉络。最后，剖析一个功能完备的邮签打印程序的开发过程，综合应用了 GDI+、ADO.NET 和 XML 技术，让读者体会如何将学到的知识应用于实践，将一个个独立的功能代码段组合为一个真实的程序（参见图 8）。

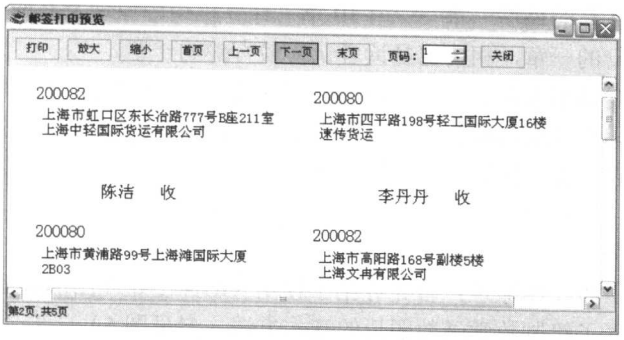

图 8　邮签打印程序

第 3 篇共有 6 章，占据了本书一半的篇幅，深入介绍了.NET 中最重要的三个技术领域：数据库、XML 和 GDI+。至此，对.NET 具体技术领域的介绍告一段落，从第 4 篇开始，转入更为专业化的领域——基于组件的编程。

笔者认为，前面所学的各种.NET 编程技术，其实只是在教读者掌握一些工具，工具本身固然要掌握，但更重要的是掌握使用这个工具去实践的方法。套句老话：

知识本身不是重要的，知道如何去应用知识才是重要的。

同样用一支毛笔，书法家写的字一字千金，而普通人的字只能戏称“涂鸦”，其差别就在工具的应用水平上。

第 4 篇主要介绍如何应用具体技术去开发实用软件的问题，正因如此，第 4 篇的名字是“精通”，掌握这部分的内容是读者走向职业程序员的必经之路。

如果说从“人”的思维转向“计算机”的思维是初学者必须要过的一个坎，那么，从“结构化”的编程方法转向“面向对象”的编程方法则是另一个坎。然而，如果读者并没有根深蒂固的结构化编程思维习惯（往往这是在大量的 C 语言编程实践中培养的），而是从本书直接学习面向对象编程的，那么第二个坎就并不存在了，从这个意义上说，没有“沉迷”C 语言可能还真有那么一点好处。

面向对象方法取代结构化方法成为软件开发的主流，是软件技术发展的必然趋势。然而，只有到可以将面向对象的思想应用在组件这个级别之上时，面向对象技术在软件开发领域的威力才能得到真正淋漓尽致的发挥。

与前一时代的组件技术——COM 相比，.NET 所提供的全新的组件化开发技术将极大地提高软件开发的效率。用“搭积木”的方法构造软件系统已经可以大规模地应用在软件系统的开发中，几乎第四部分的所有内容，都是围绕着“组件化开发”这一主题而展开的。

首先，在第 11 章“面向对象软件编程基础”中介绍面向对象编程的基本理论与方法，并以一个简单的实例来展现一个普通的小程序是怎样一步一步地走向组件化道路的（参见图 9）。

这个程序功能虽然简单，但在简陋的外表之下却蕴涵着组件化开发的基本思想。

接着，介绍软件动态链接技术的来龙去脉，这是构建.NET 强大组件技术框架的基石，让读者了解到软件技术发展背后，总有一些不变的思想在起作用。

本章的最后，介绍以图形化来描述面向对象软件系统的 UML（Unified Modeling Language，统一建模语言）。UML 已成为国际标准，是当今软件工程师所必须掌握的基本知识。可以这么说，不知道 UML，看不懂 UML 图，就不是一个合格的软件工程师！

在第 11 章介绍完了组件化开发的基础知识之后，从第 12 章开始，正式介绍.NET 的组件化开发技术。

首先，介绍什么是软件组件，如何把组件装配成一个完整的软件系统，这种开发方式为何被称为是软件开发方法的一场变革。

接着，介绍如何在.NET 中开发可以复用的软件组件，以一个支持自动匹配的文本框 SuperTextBox 的开发全过程为例介绍.NET 自定义控件的开发技术（参见图 10）。

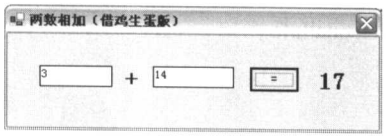

图 9　一个简单的基于组件的小程序

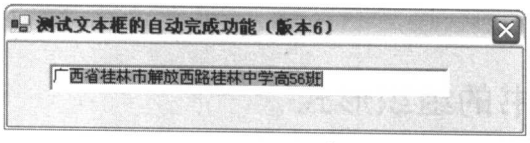

图 10　具有自动完成功能的文本框控件

详细介绍这个自定义控件的开发过程，主要目的是想让读者明白如何重构一个已有程序，以及创建自己的组件库的过程。一旦创建好了自己的组件库，再开发新系统就成了复用已有的组件，开发部分新组件，再将新旧组件装配成新系统的过程。这种基于组件的软件开发方式，是对以编写大量代码为主要特征的传统开发方法的重大革新。

组件化开发并不是微软公司的独创，但在.NET 平台中，支持混合语言开发组件，支持组件动态插拔，使.NET Framework 成为了一个支持组件化开发的优秀软件平台。因此，本章详细介绍了这两种技术，这两种技术常用于团队开发，掌握它们，您就可以与其他人合作，开发出结构复杂、功能强大的软件系统。

如果读者能掌握这些技术，那么可以理直气壮地说，您已经知道面向对象是怎么回事了！

第 12 章介绍了许多零散的组件化开发技术，第 13 章则介绍如何将这些零散的技术组合起来，使用组件化开发方法开发出一个工具软件——PersonalInfo（参见图 11）。

PersonalInfo 是一个功能强大而且实用的工具软件，我写作这本书时，所需的大部分资料及安排写作计划等工作都是由它来管理的。

PersonalInfo 这个程序几乎用到了本书介绍的全部技术，源代码总数接近一万行！

本章从系统分析阶段开始，介绍单个组件的开发，如何装配软件组件，最后介绍真实的软件