汇主流编程语言 精选典型编程案例，聚宝贵编程经验 揭示编程技术诀窍

电脑编程技巧与维护

2009年合订本

精华版

《电脑编程技巧与维护》杂志社 编著

高手解读，编程热点技术

实例导航，引领编程捷径

内容精编，荟萃编程技巧

代码移植，编程方便快捷

二次加工，全 新 全 品

中国水利水电出版社
www.waterpub.com.cn

《电脑编程技巧与维护》
2009年合订本
（精华版）

《电脑编程技巧与维护》杂志社　编著

中国水利水电出版社
www.waterpub.com.cn

内 容 提 要

《电脑编程技巧与维护》2009年合订本（精华版）是在保留杂志原有风格的基础上，精心选编了2009年上半月刊12期的典型编程开发范例，汇集众多编程高手项目开发和应用的经验和编程技巧，精心加工后形成的全新编程产品。所选跟高手学编程、编程语言、数据库、网络与通信、图形图像处理、游戏编程、计算机安全与维护、编程疑难问题解答8篇近百个编程案例，都是从实际项目提炼出的，其内容有深度、思路有新意、讲解深入浅出，编程技巧新颖实用，构思巧妙，编程技术覆盖面广。案例中的程序源代码都经过上机调试通过，随书附带一张本书所有核心源代码光盘，为程序开发人员移植代码和应用编程带来了方便。

本书以案例导航、高手解读、答疑解惑的方式，诠释编程热点技术，传授编程经验技巧，引领编程捷径。全书既讲究内容的系统性、深入性、专业性、权威性和实用性，同时兼顾轻松、通俗易懂、时效性强的特点。

本书可作为电脑编程爱好者、软件开发人员、专业计算机系统维护人员和专业程序员进行项目开发、项目设计的参考书，也可作为软件从业人员及编程爱好者的珍藏宝典。

图书在版编目（CIP）数据

电脑编程技巧与维护 : 2009年合订本 : 精华版 / 《电脑编程技巧与维护》杂志社编著. -- 北京 : 中国水利水电出版社, 2010.1
ISBN 978-7-5084-6954-6

Ⅰ. ①电… Ⅱ. ①电… Ⅲ. ①程序设计②微型计算机—维修 Ⅳ. ①TP311.1②TP360.7

中国版本图书馆CIP数据核字(2009)第205372号

策划编辑：杨庆川　责任编辑：杨元泓　封面设计：李　佳

书　名	《电脑编程技巧与维护》2009年合订本（精华版）
作　者	《电脑编程技巧与维护》杂志社　编著
出版发行	中国水利水电出版社 （北京市海淀区玉渊潭南路1号D座　100038） 网址：www.waterpub.com.cn E-mail：mchannel@263.net（万水） sales@waterpub.com.cn 电话：（010）68367658（营销中心）、82562819（万水）
经　售	全国各地新华书店和相关出版物销售网点
排　版	北京万水电子信息有限公司
印　刷	北京蓝空印刷厂
规　格	184mm×260mm　16开本　25.75印张　638千字
版　次	2010年1月第1版　2010年1月第1次印刷
印　数	0001—6000册
定　价	46.00元（赠1CD）

凡购买我社图书，如有缺页、倒页、脱页的，本社营销中心负责调换

前　言

《电脑编程技巧与维护》是为从事电脑编程、系统应用开发人员创办的专业性和实用性都很强的技术刊物。自 1994 年创刊 15 年来，始终以“实用第一，智慧密集”为宗旨，坚持“质量第一”、“读者第一”的原则，为广大的电脑编程爱好者、软件开发人员和专业计算机系统维护人员提供第一手的技术资料、编程技巧和维护经验；紧紧跟踪计算机软硬件技术发展和应用趋势，不断求变创新，针对软件开发过程中许多关键技术问题，着重提供各类解决方案和项目典型开发范例，在业内获得一致好评，是广大编程爱好者的首选刊物。在栏目内容上，选题覆盖面广，涉及技术领域宽、信息量大，帮助程序员开阔视野；在技术水平上，始终把握计算机技术发展的大方向，提供先进、详尽、准确的技术指导，并在长期的工作中与国际型大公司建立了良好的合作关系，为读者提供最新、最全的实用信息；在实用性上，稿源来自专业开发和维护人员的实践经验，以及普通书籍难以获得的编程经验、体会与技巧。

《电脑编程技巧与维护》从 2009 年开始，经国家新闻出版总署批准，由月刊改为半月刊。上半月刊的办刊宗旨、读者对象、栏目设置、刊物内容保持原刊风貌，仍以电脑编程案例解析为主题；下半月刊是原《电脑编程技巧与维护》月刊内容的延伸与发展，汇集电脑编程相关的学术论文和研究报告，展示电脑编程应用开发研究的新进展、新成果、新方法。

2009 年《电脑编程技巧与维护》杂志社与中国水利水电出版社共同策划和倾力打造出版了《电脑编程技巧与维护》2009 年合订本（精华版），作为 2010 年的一份礼物献给广大的读者。

2009 年《电脑编程技巧与维护》合订本（精华版）的内容由《电脑编程技巧与维护》全年上半月刊重点栏目第 1 期至第 12 期中精选的典型、实用、精彩的近百个典型开发和应用编程案例组成，分成跟高手学编程、编程语言、数据库、网络与通信、图形图像处理、游戏编程、计算机安全与维护、编程疑难问题解答 8 篇，涵盖了当前主流编程语言 VC++、C#、Java、ASP.NET 等。合订本精华版有如下几个显著特点：

1．案例经典，内容精华

合订本精华版中的实例都是作者从实际项目提炼出的开发范例，稿源来自专业开发和系统维护人员的原创，其中不少文章的作者是业界资深程序员和技术专家，内容有深度、思路有新意、讲解深入浅出，编程技巧新颖实用，构思巧妙，编程技术覆盖面广，适用于各类编程人员，是编程人员学习编程时从有关编程书籍和网络上很难得到的学习参考资料。

2．高手解读，精彩展现

合订本精华版汇集了众多编程高手项目开发和应用的经验和编程技巧，实例讲解先给出设计目标，然后介绍实现目标的基本思路和实现方法，最后详细给出其核心程序源代码，对其核心源代码进行清晰的注释解读，并给出程序的运行结果。对于编程中的疑难问题进行了深入解答，全面展示编程技术与诀窍。

3．二次加工，全新产品

合订本精华版保留了杂志的原有风格，但不是 12 期内容的简单相加，是经过二次加工后形

成的全年上半月刊全新编程产品。为方便读者阅读，合订本精华版的每一篇内容都按照编程语言的类别重新精心编排。

4．代码移植，方便快捷

合订本精华版中的每个实例的程序源代码都经过上机调试通过，并随书附带一张本书所有核心源代码的光盘，使程序开发人员移植代码和应用编程时方便快捷。

合订本精华版既讲究内容的系统性、深入性、专业性、权威性和实用性，同时兼顾轻松、通俗易懂、时效性强的特点。

本书可作为电脑编程爱好者、软件开发人员、专业计算机系统维护人员和专业程序员进行项目开发、项目设计的参考书，也可作为软件从业人员及编程爱好者的珍藏宝典。

《电脑编程技巧与维护》杂志社

2009 年 12 月

目　录

第一篇　跟高手学编程

1.1　VC++.NET 访问数据库，编程实现存取股价数据

范晓平　方　阳

程序设计语言与数据库是两种不同的软件开发平台。VC++.NET 本身并不包含数据库。如果 VC++.NET 要修改或使用其他数据库的数据，必然涉及两种不同开发平台之间的数据交流，即 VC++.NET 访问数据库。

本讲将通过一个应用程序实例来讲解 VC++.NET 访问数据库的编程方法及实现过程，这个实例程序还将贯通到本篇后续的 1.2 和 1.3 两讲中去。

1　ADO.NET

要从应用程序中方便地访问数据库，必须经过二者之间的接口。在过去的 Microsoft Visual Studio 平台上，已经拥有许多数据库访问技术，主要包括：

（1）ODBC API（Open DataBase Connectiyity）。

（2）MFC ODBC（Microsoft Foundation Classes ODBC）。

（3）DAO（Data Access Object）。

（4）OLE DB（Object Link and Embedding DataBase）。

（5）ADO（ActiveX Data Object）。

其中，ADO 是 Visual Studio 平台上首选的数据库访问技术。但是，自从 Visual Studio.NET 平台引入了新的数据库访问技术——ADO.NET 以后，改变了这种情况。ADO.NET 解决了 ADO 无法满足的需求，使数据库操作变得更简单，代替了 ADO 的首选地位。

1.1　优势

与 ADO 比较，ADO.NET 提供了以下若干好处：

1.1.1　互操作性

XML 是网络中传输数据集的一项工业标准。在 ADO.NET 中，传输数据的格式采用 XML。

使用 ADO.NET，数据从数据存储区移动到数据集以及从数据集移动到各种组件，都用 XML 格式。同样，如果需要保持数据（例如保持到文件中），则将其存储为 XML。如果有 XML 文件，则可以像使用任何数据源一样使用它，并从它创建数据集。

因此，应用程序中的数据组件可以与其他任何应用程序中的其他任何组件交换数据，只要该组件理解 XML。而许多应用程序被编写为可理解 XML，这为异类应用程序间的互操作性提高了空前高的水平。

1.1.2　可维护性

应用程序部署到系统以后，往往还需要做适度的更改。例如，当已部署的应用程序越来越受用户欢迎时，增加的性能负荷可能需要进行结构更改。随着已部署的应用程序服务器上的性能负荷的增长，系统资源会变得不足，并且响应时间或吞吐量会受到影响。面对这样的问题，软件设计者可以选择将服务器上的业务逻辑处理和用户界面处理划分到单独计算机的单独层上。应用程序服务器层替换为两层以后，缓解了系统资源缺乏。

这个问题并不是要在开发之初设计三层应用程序，相反，它是要在应用程序部署以后增加层数。如果原始应用程序使用 ADO，这种实质上的更改是很困难的。但是如果原始应用程序使用数据集以 ADO.NET 实现，则该转换很容易进行。当用两层替换单层时，涉及这两层之间交换信息。由于这两层可以通过 XML 格式的数据集传输数据，所以通信相对较容易。

1.1.3　可编程性

ADO.NET 以不同方式封装数据访问功能，帮助加快编程速度并减少犯错几率。

1.1.4　性能

对于不连接的应用程序，ADO.NET 数据集提供的性能优于 ADO 不连接的记录集。当使用 COM 封送在层间传输不连接的记录集时，会因将记录集内的值转换为 COM 可识别的数据类型而增大处理开销。在 ADO.NET 中，这种数据类型转换则没有必要。

总之，ADO.NET 在功能、性能上都超越了 ADO，成为.NET 平台上首选的数据库访问技术。

1.2　结构

ADO.NET 实际上是一组向 .NET 程序员公开数据访问服务的类。应用程序通过 ADO.NET 来连接到这些数据源，并检索（SELECT）、操作（Alert/Delete）和

更新（Update）数据。

设计 ADO.NET 组件的目的是为了从数据操作中分解出数据访问。

ADO.NET 包含两个核心组件。

（1）DataSet。

（2）.NET Framework 数据提供程序。DataSet 是 ADO.NET 的断开式结构的核心组件。DataSet 的设计目的很明确：实现独立于任何数据源的数据访问。因此，它可以用于多种不同的数据源，用于 XML 数据，或用于管理应用程序的本地数据。DataSet 包含一个或多个 DataTable 对象的集合，这些对象由数据行和数据列以及主键、外键、约束和有关 DataTable 对象中数据的关系信息组成。

.NET Framework 数据提供程序包括一组对象：Connection、Command、DataReader 和 DataAdapter。其组件的设计目的相当明确：实现数据操作和对数据的快速、只进、只读访问。Connection 对象提供与数据源的连接。Command 对象能够访问用于返回数据、修改数据、运行存储过程以及发送或检索参数信息的数据库命令。DataReader 从数据源中提供高性能的数据流。最后，DataAdapter 提供连接 DataSet 对象和数据源的桥梁。DataAdapter 使用 Command 对象在数据源中执行 SQL 命令，以便将数据加载到 DataSet 中，并使对 DataSet 中数据的更改与数据源保持一致。

.NET Framework 提供了四个 .NET Framework 数据提供程序：SQL Server .NET Framework 数据提供程序、OLE DB .NET Framework 数据提供程序、ODBC .NET Framework 数据提供程序和 Oracle .NET Framework 数据提供程序。

ADO.NET 结构如图 1 所示。

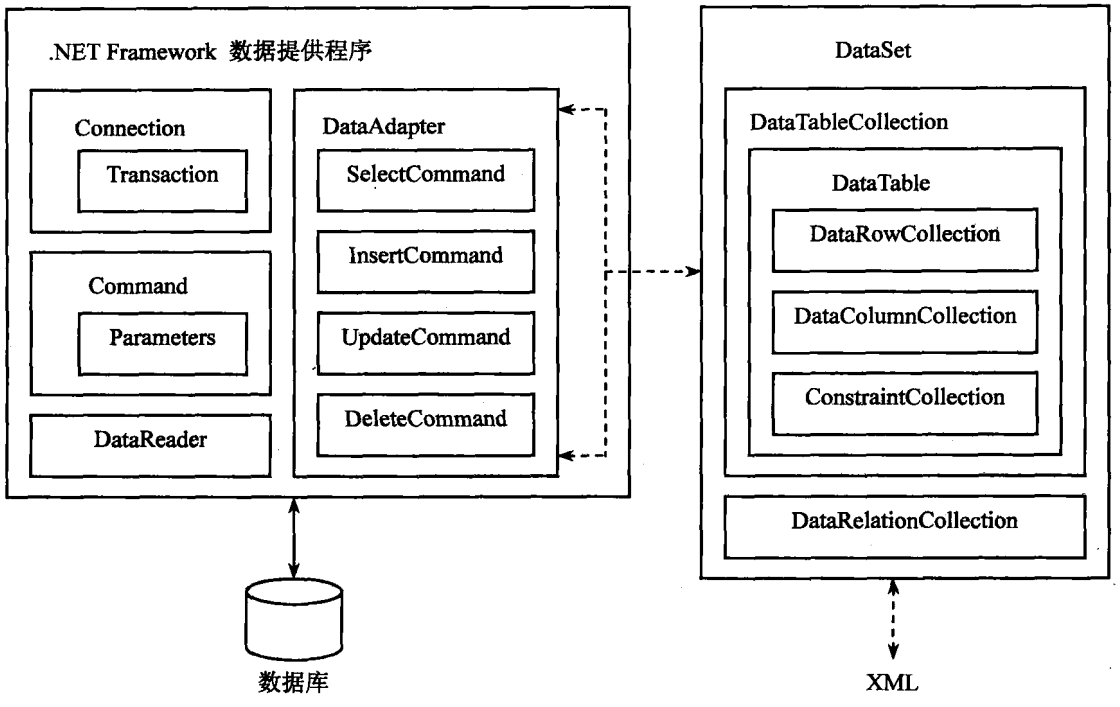

图 1 ADO.NET 结构

DataSet 的设计能够使用 XML Web Services 方便地通过 Web 将数据传输到客户端，并允许使用 .NET 远程处理服务在 .NET 组件之间封送数据。还可以通过这种方式对强类型的 DataSet 进行远程处理。

要注意的是，DataTable 对象也可以与远程处理服务一起使用，但不能通过 XML Web Services 进行传输。

ADO.NET 类在 System.Data.dll 中，并且与 System.Xml.dll 中的 XML 类集成。当编译使用 System.Data 命名空间的代码时，应引用 System.Data.dll 和 System.Xml.dll。

2 实例程序说明

2.1 需求分析

该软件是为一位股民设计的。他参与股市已有十多年。多年的炒股经历使他积累了丰富独特的股市中长期分析经验。这些经验既有技术分析，也有基本分析。对技术分析来说，有对现有图表的改造、组合，也有自己的发明。对基本分析来说，针对不同时期有不同的分析

指标，还有对多指标的综合。显然，运用这些经验，需要大量的数学计算，靠人脑、手算不但十分繁琐还容易出错。

他希望利用电脑来代替繁琐的手工计算，根据他的分析方法编制一个股市分析软件，辅助他个人进行股市中长期走势分析。

首先，他要求软件：

（1）录入、编辑数据。因为软件用作中、长期分析，所以数据不必与股市同步，允许时间滞后一些。

（2）绘股价走势图。

（3）绘成交量分布图。

（4）制股市行情表。

随后，将他的分析方法添加到软件中，像搭积木一样逐步完善。

分析他的要求，软件在上述开发中应具有以下功能需求：

（5）数据录入、编辑界面。数据包括日期、开盘价、收盘价、成交量、最高价和最低价。

（6）利用日期、开盘价、收盘价、最高价和最低价绘制股价的K线图。

（7）利用日期和成交量绘制条形图。

（8）利用所有数据制作股市行情表。

此外，为了方便软件后续开发，软件还应该留有可扩充接口。

2.2　选择开发环境

软件开发平台选择VC++.NET 2005，数据库服务器选择SQL Server 2000。

VC++升级到VC++.NET以后，两者最大的区别是它们依赖的框架不同，前者是MFC（Microsoft Foundation Classes），后者是.NET Framework。

.NET Framework比MFC庞大得多，不仅支持VC++.NET，还支持VB.NET、Visual C#和Visual J#。同时，.NET Framework由于受公共语言运行库（Common Language Runtime，CLR）支持，为软件设计人员提供了托管应用程序开发环境。可以说，CLR是.NET Framework的最大亮点。尽管如此，微软仍然在.NET Framework中为非托管应用程序留有一席之地，MFC其中就占据一角。.NET Framework环境如图2所示。

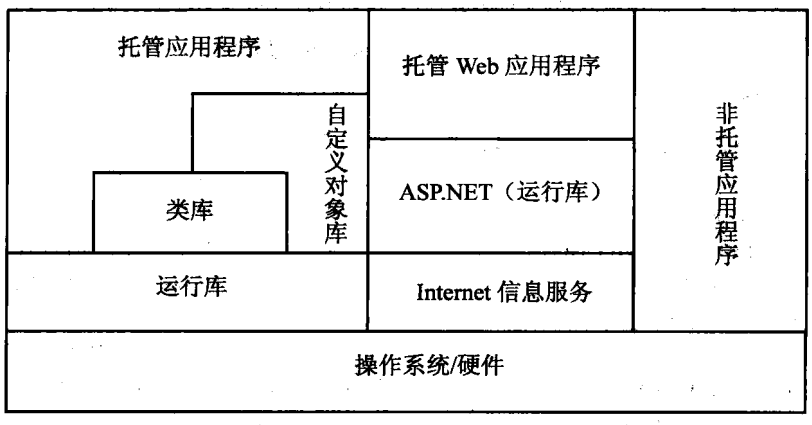

图2　.NET Framework 环境

使用CLR有许多好处，其中之一是可以在应用程序中使用Windows窗体，这为设计应用程序用户界面提供了很大的方便。因此，实例程序将选择CLR类型的Windows窗体应用程序作为项目模板。

2.3　程序构成

（1）程序有一个主窗体，这是用户进入系统的第一个界面，也是连接其他窗体的起点。

（2）对应其他功能，还应有其他窗体。

1）数据录入、编辑使用一个窗体。

2）股价的K线图和成交量的条形图是配合使用的，通常分别绘在一个图的上下位置，所以用一个窗体就可以了。

3）股市行情表本来一个窗体也可以了，但是作为演练，可以将单页报表和分页报表分开设计，这样讲解会更清楚一些。因此股市行情表可以规划为两个窗体。

此外，在主窗体上，还要加上菜单。考虑到软件可扩充，可以安排三个菜单项：数据、技术分析和基本分析。在技术分析中，暂时添加绘图和制表两个菜单命令，在制表菜单命令下添加两个子菜单：单页报表和分页报表。

3　准备数据

数据是绘图、制表和对股市走势进行其他分析的原料。需要持久保持数据时，用数据库来保存和管理。

实例程序中需要持久保持的数据是股市的基本信息，包括日期、开盘价、收盘价、成交量、最高价和最

低价。

为这些信息创建一个数据库 Stock。在数据库中创建一个数据表 StockList。

3.1 StockList 表的数据结构

数据表 StockList 用来保存股市的基本信息，其数据结构如表 1 所示。

表 1 StockList 表的数据结构

字段名称	数据类型	长度	精度	小数位数	说明
日期	smalldatetime	4	0	0	主键
开盘价	decimal	5	7	2	
收盘价	decimal	5	7	2	
成交量	int	4	10	0	
最高价	decimal	5	7	2	
最低价	decimal	5	7	2	

为便于调试程序，可以通过 SQL Server 的企业管理器在 StockList 表中预先输入一些调试数据。实际数据由用户通过实例程序的数据界面录入、编辑。

3.2 创建 StockList 表

创建 StockList 表也可以使用 SQL 代码创建。表 StockList 要求下面的 DDL 语句：

```
CREATE TABLE StockList (
  日期 smalldatetime PRIMARY KEY NOT NULL,
  开盘价 numeric(7, 2) NOT NULL,
  收盘价 numeric(7, 2) NOT NULL,
  成交量 int NOT NULL,
  最高价 numeric(7, 2) NOT NULL,
  最低价 numeric(7, 2) NOT NULL
)
GO
```

4 实例程序编程

本节通过实例程序编程，讲解应用程序基于 ADO.NET 访问数据库的整个实现过程。

4.1 创建框架应用程序

（1）启动 Visual Studio。

（2）在“文件”菜单上，单击“新建”，然后单击“项目”。

（3）在“项目类型”窗格中，选择 Visual C++节点中的 CLR，然后在“模板”窗格中选择“Windows 窗体应用程序”。

（4）键入项目的名称：StockMrktAnls。同时键入一个要保存项目的目录，或者浏览到要保存项目的目录。单击“确定”创建框架应用程序。

创建框架应用程序后，即可添加所需的功能。

4.2 设置主窗体属性

在框架应用程序中，Visual Studio 已经自动创建了一窗体 Form1。利用这个窗体作为应用程序的主窗体。为此，对窗体属性作以下修改：

（1）将文件名 Form1.h 改为 StockMrktAnls.h，并替换项目所有文件中出现的 Form1.h。

（2）将 Text 属性改为“股市分析软件”。

（3）适当调整窗体的大小。

这些修改对应用程序的功能并不是必要的，只是为了统一程序中文件名称命名的约定或点缀一下窗体的外观。

4.3 添加标签

在主窗体中添加一个标签 label，将 Text 属性改为“股市分析软件”，并调整其在窗口中的位置。再根据自己的喜好适当选择字体名称、大小、前景颜色。

4.4 创建菜单

菜单可以将应用程序中功能用合乎逻辑并且易于查找的方式进行排列。通过菜单，用户可以非常方便地在系统中查找到自己所需的功能。

现在来为主窗体添加菜单。

创建菜单可以使用菜单编辑器。可以直接使用与最终的应用程序中的菜单栏十分相似的菜单栏来创建和编辑菜单。

（1）打开“工具箱”的“菜单和工具栏”选项卡，将 menuStrip 控件拖放到主窗体上。

（2）单击菜单栏上的“请在此处输入”矩形。键入新菜单的名称“数据”。

键入的文本同时出现在菜单编辑器和“属性”窗口的 Text 框中。可以在这两个位置中的任何一个编辑新菜单的属性。

给菜单栏上的新菜单提供名称后，新项框移到右边（以允许添加其他菜单），另一个新项框在第一个菜单的下面打开，以便可以向其中添加菜单命令。

（3）在“数据”菜单右边新项框中键入第二个菜单的名称“技术分析”。

（4）在“技术分析”菜单下面的新项框中添加两个菜单命令“绘图”和“制表”。

（5）在“技术分析”菜单右边新项框中键入第三个菜单的名称“基本分析”。这个菜单在这次设计中不用，留给以后系统扩充时使用。

完成以上步骤后，现在主窗体布局应该如图 3 所示。

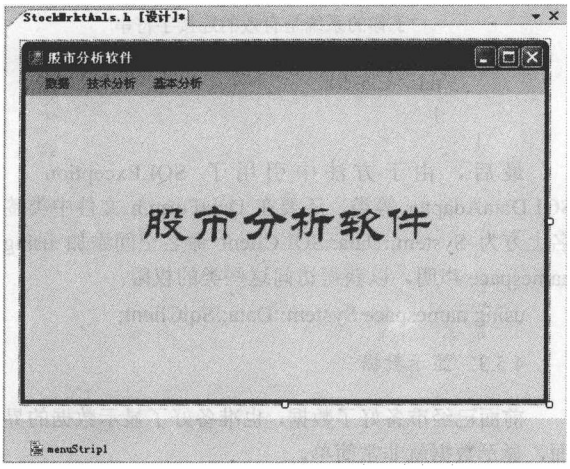

图 3　主窗体布局

4.5　访问数据库

从实例程序访问数据库主要解决三个问题：如何从数据库取数据；当从数据库取到数据以后，如何将数据显示给用户；如果用户修改过数据，如何将修改保存到数据库。

4.5.1　显示数据准备工作

当从数据库取到数据以后，选择 DataGridView 控件来显示数据。

DataGridView 控件提供了一种强大而灵活的以表格形式显示数据的方式。可以使用 DataGridView 控件来浏览数据，也可以对数据进行编辑（修改、删除或增加）。

DataGridView 控件可以显示和编辑来自多种不同类型的数据源的表格数据。将数据绑定到 DataGridView 控件非常简单和直观，在大多数情况下，只需设置 DataSource 属性即可。在绑定到包含多个列表或表的数据源时，只需将 DataMember 属性设置为指定要绑定的列表或表的字符串即可。

DataGridView 控件功能十分强大，使用起来非常方便，是 Windows 窗体应用程序中显示表格数据的首选。

通常情况下，DataGridView 控件并不直接绑定到数据源，而是绑定到 BindingSource 组件，再将 BindingSource 组件绑定到其他数据源或使用业务对象填充该组件。如图 4 所示。

图 4　DataGridView 控件、BindingSource 组件与数据之间关系示意图

由于 BindingSource 组件可以绑定到各种数据源，并可以自动解决许多数据绑定问题，所以选择 BindingSource 组件作为 DataGridView 控件的数据源。

最后，为项目添加一个窗体来容纳 DataGridView 控件，用作录入、编辑数据作的界面。当用户从主窗体选择菜单“数据”以后，将显示这个窗体。

（1）窗体的名称为 DataForm。

（2）窗体的 Text 属性为“数据”。

（3）保持 DataGridView 控件的 Name 属性的默认值 dataGridView1 不变。

（4）保持 BindingSource 组件的 Name 属性的默认值 bindingSource1 不变。

（5）适当调整 DataGridView 控件的位置和大小，并相应调整窗体的大小。

窗体的布局（设计区域）如图 5 所示。

4.5.2　从数据库取数据

为了从数据库取数据，在 DataForm 类中编写一 GetData 方法。

该方法对一个 SqlDataAdapter 组件进行初始化，并使用该组件填充 DataTable。然后，将 DataTable 绑定到 BindingSource 组件。

在对 SQLDataAdapter 组件进行初始化时，使用了两个参数 selectCommand 和 connectionString。

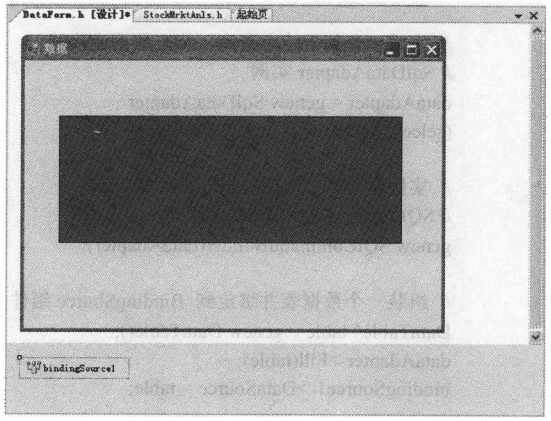

图 5　DataForm 窗体的布局

（1）selectCommand 是一个 SQL 命令的字符串，例如 SELECT（查询）、UPDATE（更新）或 DELETE（删除），字符串必须符合 SQL 语法规范。该参数由这个方法的调用者传入。

（2）connectionString 是一个连接字符串，用于连接数据库。它的设置随数据库不同而不同。对于这个程

序使用的数据库而言，其设置是：

"Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=Stock; Data Source=localhost"

连接字符串的设置也有严格的规定，其中参数的含义如下：

（1）Data Source　要连接的 SQL Server 实例的名称或网络地址。

（2）Persist Security Info　如果将该关键字设置为 true 或 yes，允许在打开连接后，从连接中获得涉及安全性的信息（包括用户标识和密码）。如果将 Persist Security Info 保持为 false 或 no，则不允许从连接中获得涉及安全性的信息。

（3）Initial Catalog　数据库的名称。

（4）Integrated Security　当设置为 false 时，将在连接中指定用户 ID 和密码；为 true 时，将使用当前的 Windows 账户进行身份验证。

此外，SQLDataAdapter 类型的 dataAdapter 变量是作为 DataForm 类中的一个成员变量被定义的：

```
private: SQLDataAdapter^ dataAdapter;
```

GetData 方法的代码如下：

```
private: void GetData(String^ selectCommand)
{
    try
    {

      // 指定一个连接字符串。这好比选择路径的方向，
      // 如果方向不对，则无法建立通路。对于不同数
      // 据库，其连接字符串设置也不同
      String^ connectionString = "Integrated Security=
        SSPI;Persist Security Info=False;" +"Initial
        Catalog=Stock;Data Source=localhost";

      // 生成一个基于特定查询、特定数据连接的
      // SqlDataAdapter 实例
      dataAdapter = gcnew SqlDataAdapter
      (selectCommand, connectionString);

      // 实例化 SQLCommandBuilder 来产生
      // SQL 更新、插入和删除命令
      gcnew SQlCommandBuilder(dataAdapter);

      // 组装一个数据表并绑定到 BindingSource 组件
      DataTable^ table = gcnew DataTable();
      dataAdapter->Fill(table);
      bindingSource1->DataSource = table;

      // 调整 DataGridView 控件的列数以适合新
      // 近加载的数据表
      dataGridView1->AutoResizeColumns(
          DataGridViewAutoSizeColumnsMode::
          AllCellsExceptHeader);
    }
    catch (SQlException^)
    {
      MessageBox::Show("为了运行实例程序，用一个对"
        +"于你的系统是有效的连接字符串"
        +"代替 connectionString 变量的值。");
      this->Close();
      }
  }
```

最后，由于方法中引用了 SQLException、SQLDataAdapter 等类，还要在 DataForm.h 文件中类签名上方为 System::Data::SQLClient 命名空间添加 using namespace 声明，以获得访问这些类的权限：

```
using namespace System::Data::SqlClient;
```

4.5.3　显示数据

前面已经准备好了数据，也准备好了显示数据的界面，显示数据就非常简单。

在窗体 DataForm 中的 Load 事件中，将 DataGridView 控件绑定到 BindingSource 组件，同时以“select * from StockList”字符串作为参数调用 GetData 方法。

“select * from StockList”是 SQL 查询命令，它被传入 GetData 方法中，与连接字符串一起作为参数，实例化由 SQLDataAdapter 声明的 dataAdapter 变量。

DataForm_Load 代码如下：

```
private: System::Void DataForm_Load(System:: Object^
sender, System::EventArgs^  e)
{
      // 绑定 DataGridView 到 BindingSource
      // 组件并且从数据库加载数据。
      dataGridView1->DataSource =
      bindingSource1;
      GetData("select * from StockList");
}
```

至此，可以在窗体 DataForm 中显示数据了。还要解决的问题是：从主窗体的“数据”菜单显示窗体 DataForm。

在窗体设计器中打开 StockMrktAnls.h 文件，然后双击“数据”菜单，即在代码编辑器中打开 StockMrktAnls.h 代码文件，并自动生成了数据 ToolStripMenuItem_Click 事件的过程体。

在 ToolStripMenuItem_Click 过程中添加以下代码：

```
DataForm^ datafrm=gcnew DataForm();
datafrm->Show();
```

最后，在 StockMrktAnls.h 代码文件的顶部添加一行代码，将 DataForm 窗体的头文件包含进来：

```
#include "DataForm.h"
```

4.5.4　保存数据

现在，虽然可以显示数据，也可以对数据进行修改，但是修改的结果只显示在 DataGridView 控件中，当关闭窗体后，修改并没有保存到数据库。

为了保存修改，还要做以下工作：

（1）在窗体 DataForm 中添加一个 Button 控件，名

称属性设置为 submitButton，Text 属性设置为“保存数据”。

（2）编写 Button 控件的 Click 事件代码：

```
private: System::Void button1_Click(System:: Object^  sender,
System::EventArgs^  e)
{
    try
        {
            // 用用户的修改更新数据库。
            dataAdapter->Update((DataTable^)
                bindingSource1->DataSource);
            // 关闭当前窗体。
            this->Close();
        }
        catch (SQLException^)
        {
            MessageBox::Show("日期必须唯一，每一
                行的日期不能相同。");
        }
}
```

4.5.5　将连接字符串保存在应用程序配置中

前面在讲解从数据库取数据时，提到要设置一个用于连接数据库的连接字符串。它的设置只随数据库不同而不同。对于这个实例程序来说，使用的数据库始终相同，因此连接字符串也始终相同。实例程序不仅在本讲中使用，还要在后两讲中使用。因此，可以将连接字符串保存在应用程序配置文件 app.config 中，在需要使用的时候从 app.config 中读取。

app.config 文件是一个 XML 文本文件，包含应用程序特定的一些设置。它既包含公共语言运行库读取的设置（如程序集绑定策略、远程处理对象等等），也包含应用程序可以读取的设置。

使用 app.config 文件有两大优点：

（1）减少代码量并易于维护。如果将需要重复使用的变量分散在多处代码中，不仅要重复录入，而且一旦改动则需要遍历项目中所有引用这个变量的代码。但是，如果将变量集中保存在 app.config 文件中，只需录入一次，改动也只需更改一个地方。

（2）当更改配置后，无需重新编译应用程序即可更新应用程序的某些配置数据。例如，当数据库迁移到另一个不同的服务器以后，只需修改 app.config 文件中数据库连接配置，不需要重新编译和重新部署这个应用程序就可以适应新的服务器的要求。

在本程序中，改用 app.config 文件保存连接字符串，需要对原来的编码进行一些修改和补充。操作如下：

（1）在项目中添加 app.config 文件。

在“项目”菜单上单击“添加新项”。随即显示“添加新项”对话框。选择“实用工具类别”，再选择“配置文件（app.config）”模板，然后单击“添加”。名为 app.config 的文件被添加到项目中。

再对项目进行设置。在“项目”菜单上单击“项目属性”，打开属性页，展开“生后事件”节点，在“生成后事件”添加下列设置：

命令行：type app.config > "$(TargetPath).config"

说明："Updating target's configuration file"

（2）在 app.config 文件中配置连接字符串。

打开 app.config 文件，在 Configuration 元素中添加一个 appSettings 元素（粗体字）：

```
<configuration>
  <appSettings>
    <add key="ConnectionString"
         value="Integrated
         Security=SSPI;Persist Security
         Info=False;Initial
         Catalog=Stock;Data
         Source=localhost"/>
  </appSettings>
</configuration>
```

（3）修改 DataForm.h 代码文件。

首先，将 GetData 方法中以下语句：

```
String^ connectionString = "Integrated Security=SSPI;
Persist Security Info=False;" +
"Initial Catalog=Stock;Data Source=localhost";
```

改为：

```
String^ connectionString = ConfigurationSettings:: AppSettings
["ConnectionString"];
```

然后，在文件的顶部添加对 ConfigurationSettings 的命名空间的引用：

```
using namespace System::Configuration;
```

4.5.6　运行实例程序

（1）从“文件”菜单中，选择“全部保存”。

（2）按 Ctrl+F5 键运行应用程序。

显示主窗口，如图 6 所示。

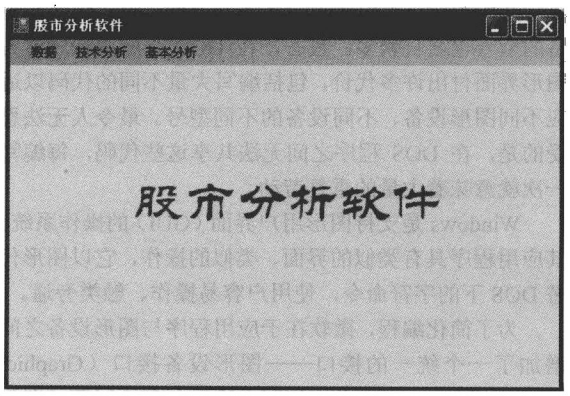

图 6　实例程序主窗口

（3）选择“数据”菜单，显示“数据”窗口，如图 7 所示。

图 7 “数据”窗口

注意：如果连接字符串设置（在 app.config 文件中）不正确，例如将数据库名称 Stock 误写为 Stoc，在选择“数据”菜单以后，将出现提示“为了运行实例程序，用一个对于你的系统有效的连接字符串代替 connectionString 变量的值。”

（4）在表格中对数据进行编辑。

如果编辑某个单元格的数据，双击单元格或按 F2 键，该单元格自动进入编辑模式。用户键入后会自动更新单元格的内容。

如果要添加新记录的行，就滚动至网格的结尾，会看到用于添加新记录的行。用户单击此行时，会向 DataGridView 控件添加使用默认值的新行。用户按 Esc 键时，此新行将消失。

如果要删除某行，首先选中整行，方法是单击行标头（该行最左边的空格），然后按 Del 键。

（5）单击“保存数据”按钮以保存所做的更改。

注意：单击“保存数据”按钮以后，如果

1）数据不符合要求，例如在股价中输入字符“a”，将显示相应的提示信息。

2）日期有重复，显示提示 “日期必须唯一，每一行的日期不能相同。”，因为日期字段在数据库中被设置为主键。

1.2 VC++.NET 图形编程——绘 K 线图和条形图

范晓平 方 阳

简单地说，图形编程就是从应用程序中将图形画到图形设备上。图形设备诸如显示器、打印机和绘图仪。

本讲将紧接第一讲采用的应用程序实例，通过绘制上证指数 K 线图和成交量条形图讲解 VC++.NET 图形编程的基本方法及实现过程。

1 Windows 图形系统结构体系

众所周知，不仅图形设备千差万别，每一种图形设备的型号也名目繁多。过去，在 DOS 下应用程序要为图形界面付出许多代价，包括编写大量不同的代码以适应不同图形设备、不同设备的不同型号。最令人无法忍受的是，在 DOS 程序之间无法共享这些代码，每编写一次就意味着大量的重复劳动。

Windows 是支持图形用户界面（GUI）的操作系统。其应用程序具有类似的界面、类似的操作，它以图形代替 DOS 下的字符命令，使用户容易操作、触类旁通。

为了简化编程，微软在于应用程序与图形设备之间增加了一个统一的接口——图形设备接口（Graphics Device Interface，GDI），如图 1 所示。GDI 屏蔽了图形设备的差异，将应用程序与不同输出设备特性相隔离，使 Windows 应用程序能够毫无障碍地在 Windows 支持的任何图形输出设备上运行。

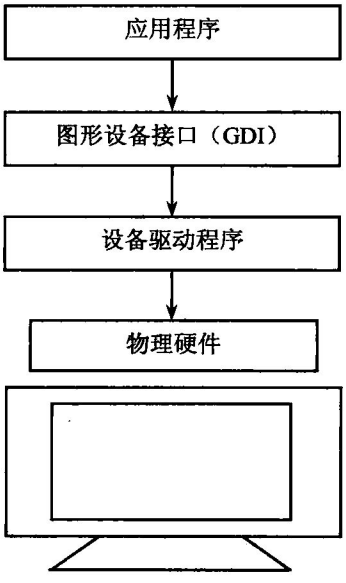

图 1 Windows 图形系统结构体系

程序员在编写图形程序时，可以不必考虑图形设备的特性，而只专注于与 GDI 打交道。程序编好以后，即使图形输出设备变换了，程序编码也无须改变。例如，能在 Epson 点式打印机上工作的程序，在不改变程序的前提下，也能在激光打印机上工作。

不仅如此，Windows 图形系统结构体系提供的 GDI 接口，适用于所有 Windows 应用程序，是 Windows 应用程序图形编程的统一接口。

2　GDI+速览

GDI 的.NET 版本叫做 GDI+。GDI+对 GDI 进行了很好的改进，并且易用性更好。

在 VC++.NET 中编写图形程序时，直接与应用程序打交道的接洽者就是 GDI+。

2.1　GDI+

GDI+是 API 通过一组部署为托管代码并向程序员公开的类。托管类接口由以下命名空间组成：

System.Drawing
System.Drawing.Drawing2D
System.Drawing.Imaging
System.Drawing.Text
System.Drawing.Printing

GDI+包含大约 60 个类、50 个枚举和 8 个结构。Graphics 类是 GDI+ 的核心功能，它是实际绘制直线、曲线、图形、图像和文本的类。

许多类与 Graphics 类一起使用。例如：

（1）Pen 类　用于绘制线条、勾勒形状轮廓或绘制其他几何形式。

（2）Brush 类　用于填充图形区域，如实心形状、图像或文本。

（3）Font 类　在绘制文本时要使用的形状。

（4）Color 结构　表示要显示的不同颜色。

2.2　确定图形的位置

GDI+图形的位置是通过坐标系来确定的。默认坐标系统的原点是在左上角，并且 x 轴指向右边，y 轴指向下边，如图 2 所示，默认坐标系统的度量单位是像素。

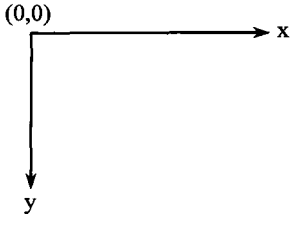

图 2　默认坐标系统

2.3　创建图形对象

所有图形都是通过 GDI+图形对象 Graphics 的有关方法来绘制的。Graphics 对象表示 GDI+ 绘图表面，是用于创建图形图像的对象。要绘制图形，需要先创建 Graphics 对象。

可以用各种方法创建 Graphics 对象。

2.3.1　从 Paint 事件中的 PaintEventArgs 引用 Graphics 对象

在窗体或控件的 Paint 事件中，接收对图形对象（作为 PaintEventArgs 的一部分）的引用。

（1）声明 Graphics 对象。

（2）分配变量来引用作为 PaintEventArgs 的一部分传递的 Graphics 对象。

（3）编写代码在窗体或控件上绘图。

下面的示例演示了如何从 Paint 事件中的 PaintEventArgs 引用 Graphics 对象。

```
private:
void Form1_Paint(System::Object ^ sender,
    System::Windows::Forms::PaintEventArgs ^ pe)
{
    // 声明一个 Graphics 对象 g，并且引用作为
    // PaintEventArgs 的一部分传递的 Graphics 对象。
    Graphics ^ g = pe->Graphics;
    // 在这里插入绘图代码。
}
```

在为控件编写绘图代码时，通常会使用此方法来获取对图形对象的引用。

2.3.2　用 CreateGraphics 方法获取对 Graphics 对象的引用

调用某控件或窗体的 CreateGraphics 方法以获取对 Graphics 对象的引用，该对象表示该控件或窗体的绘图图面。

用 CreateGraphics 方法创建 Graphics 对象的示例代码：

```
Graphics ^ g;
// 让 g 获取对 Graphics 对象的引用，
// 该对象表示该控件或窗体的绘图图面。
g = this->CreateGraphics();
```

如果想在已存在的窗体或控件上绘图，使用此方法。

2.3.3　从 Image 对象创建 Graphics 对象

可以从 Image 类派生的任何对象创建 Graphics 对象。

调用 System.Drawing.Graphics.FromImage(System.Drawing.Image)方法，提供要从其创建 Graphics 对象的 Image 变量的名称。

下面的示例演示如何使用 Bitmap 对象：

```
Bitmap ^ myBitmap = gcnew
    Bitmap("D:\\Documents and Settings\\Joe\\Pics\\
        myPic.bmp");
Graphics ^ g = Graphics::FromImage(myBitmap);
```

此方法在需要更改已存在的图像时十分有用。

2.4 绘图

遵循面向对象的普遍编程方法，处理图形包括两个步骤：

（1）创建 Graphics 对象。

（2）使用 Graphics 对象绘制线条、形状、文本或显示与操作图像。

本讲主要是绘制 K 线图和条形图，这两种图形是由线条、矩形和文本组成的。下面着重讲解线条、空心矩形、实心矩形和文本的绘制方法。

（1）绘制线条。

绘制线条调用 Graphics 对象的 DrawLine 方法，顺序传入 5 个参数：画笔（Pen）、线条左上角横坐标、纵坐标、右下角横坐标和纵坐标。

例如要从点(0,0)到(50,100)画一条线段，代码如下：

```
System::Drawing::Pen^ myPen =gcnew System::Drawing::
Pen(System::Drawing::Color::Red);
System::Drawing::Graphics^ formGraphics;
formGraphics = this->CreateGraphics();
formGraphics->DrawLine(myPen, 0, 0, 50, 100);
delete myPen;
delete formGraphics;
```

（2）绘制空心矩形。

绘制空心矩形调用 Graphics 对象的 DrawRectangle 方法，顺序传入两个参数：画笔和 Rectangle 结构。

Rectangle 结构是 GDI+提供的用于组织矩形数据的一种数据结构。它存储一组整数，共四个，表示一个矩形的位置和大小。Rectangle 结构要求 4 个 int 类型数据，依次表示矩形左上角横坐标、纵坐标、矩形的宽度和高度。

例如要画一个左上角顶点为(10,20)，宽度为 200，高度为 300 的空心矩形，代码如下：

```
private:
void DrawRectangle()
{
    System::Drawing::Pen^ myPen = gcnew System::
      Drawing::Pen(System:: Drawing::Color:: Red);
    System::Drawing::Graphics^ formGraphics;
    formGraphics = this->CreateGraphics();
    formGraphics->DrawRectangle(myPen,
        Rectangle(10, 20, 200, 300));
    delete myPen;
    delete formGraphics;
}
```

（3）绘制实心矩形。

绘制实心矩形调用 Graphics 对象的 FillRectangle 方法，顺序传入两个参数：实心画笔（SolidBrush）和 Rectangle 结构。

例如要画一个左上角顶点为(10,20)，宽度为 200，高度 300 的实心矩形，代码如下：

```
System::Drawing::SolidBrush^ myBrush = gcnew System::
    Drawing::SolidBrush(System::Drawing::Color::Red);
System::Drawing::Graphics^ formGraphics;
formGraphics = this->CreateGraphics();
formGraphics->FillRectangle(myBrush, Rectangle(10, 20,
                            200, 300));
delete myBrush;
delete formGraphics;
```

（4）绘制文本。

可以调用 Graphics 对象的 DrawString 方法在窗体上绘制文本。

DrawString 方法提供了一组重载，选用其中一种 Graphics.DrawString (String, Font, Brush, Single, Single)。它在指定位置并且用指定的 Brush 和 Font 对象绘制指定的文本字符串，参数说明如下：

第一个参数是要绘制的文本字符串。

第二个参数定义字符串的文本字体。

第三个参数确定所绘制文本的画笔（颜色和纹理）。

第四个参数确定所绘制文本的左上角的 x 坐标。

第五个参数确定所绘制文本的左上角的 y 坐标。

例如绘制文本“样板文本”、字体为 Arial（16pt）、使用实心黑色画笔、文本左上角的 x 坐标为 150、y 坐标为 50 的代码如下：

```
public:
void DrawString()
{
    System::Drawing::Graphics^ formGraphics =
        this->CreateGraphics();
    String^ drawString = "样板文本";
    System::Drawing::Font^ drawFont =
        gcnew System::Drawing::Font("Arial", 16);
    System::Drawing::SolidBrush^ drawBrush = gcnew
    System::Drawing::SolidBrush(System::Drawing::
                                Color::Black);
    float x = 150.0F;
    float y = 50.0F;
    System::Drawing::StringFormat^ drawFormat =
        gcnew System::Drawing::StringFormat();
    formGraphics->DrawString(drawString, drawFont,
                             drawBrush, x, y);
    delete drawFont;
    delete drawBrush;
    delete formGraphics;
}
```

注意：不能在 Load 事件处理程序中调用以上绘制图形或文本的方法。同时，如果已调整该窗体的大小或者其他窗体遮蔽了该窗体，也不会重绘所绘制的内容。这是因为当窗体加载或者窗体（控件）重绘时，Graphics 对象就被清空了。若要自动重绘内容，应该重写 Paint 事件处理程序。

3 绘图

本节结合实例详细讲解图形编程方法。绘图数据取