Microsoft

微软院校课程系列教材

高级编程技术

微软公司 著

高等教育出版社
HIGHER EDUCATION PRESS

微软院校课程系列教材

高级编程技术

微软公司 著

高等教育出版社

图书在版编目（CIP）数据

高级编程技术 / 微软公司著. —北京：高等教育出版社，2005.3

ISBN 7-04-016185-0

I. 高… II. 微… III. 程序设计 IV. TP311.1

中国版本图书馆 CIP 数据核字（2005）第 013089 号

策划编辑 尹 洪　**责任编辑** 叶 波　**封面设计** 张 楠　**责任绘图** 朱 静
版式设计 史新薇　**责任校对** 王 雨　**责任印制** 宋克学

出　版　高等教育出版社
社　址　北京市西城区德外大街 4 号
邮政编码　100011
总　机　010-58581000

印　刷　北京中科印刷有限公司

开　本　787×1092　1/16
印　张　25
字　数　600 000

购书热线　010-58581118
免费咨询　800-810-0598
网　址　http://www.hep.edu.cn
　　http://www.hep.com.cn

版　次　2005 年 3 月第 1 版
印　次　2005 年 3 月第 1 次印刷
定　价　80.00 元

物料号 16185-00

目　录

第 1 章　高级开发概述……1
1.1　数据存储与访问……2
1.2　多线程编程技术……2
1.3　组件化开发技术……3
1.4　XML……3
1.5　远程调用与 XML Web Service……3
1.6　文件 I/O 流……4
1.7　智能移动设备……4
1.8　虚拟执行环境……4
1.9　开发技术总结……5
1.10　架构和设计模式……5
第 2 章　组件化开发……7
2.1　组件化开发概述……7
2.1.1　面向对象技术……9
2.1.2　面向对象开发和组件化开发……10
2.1.3　组件化开发优势……10
2.1.4　组件化开发方法……11
2.2　创建组件……13
2.2.1　使用命名空间和声明类……13
2.2.2　创建类的实现……14
2.2.3　实现结构化异常处理……14
2.2.4　创建属性……16
2.2.5　编译组件……16
2.3　创建控制台客户端程序调用组件……16
2.3.1　使用类库……16
2.3.2　实例化组件……17
2.3.3　调用组件……17
2.3.4　生成客户端应用程序……18
2.4　创建 Web 客户端应用程序调用组件……18
2.4.1　创建 Web 应用程序……18
2.4.2　编写 Page_Load 事件处理程序……19
2.5　应用程序部署介绍……21
2.5.1　虚拟执行环境……21
2.5.2　在虚拟执行环境中编译和运行应用程序……22
2.5.3　部署的基本概念……28
2.5.4　简单应用程序……29
2.5.5　组件化应用程序……29
2.5.6　配置和分发……29
2.6　应用程序部署方案……30
2.6.1　简单应用程序……30
2.6.2　组件化应用程序……32
2.6.3　指定私有程序集路径……34
2.6.4　两种程序集、两种部署……35
2.6.5　强名称程序集……36
2.6.6　部署共享组件……39
2.6.7　版本化程序集……40
2.6.8　创建强名称程序集的多个版本……40
2.6.9　绑定策略……42
2.6.10　部署多版本的强名称程序集……42
2.6.11　打包和部署工具……44
2.7　参考资源……46
习题……46
第 3 章　XML 技术……48
3.1　XML 概述……48
3.1.1　XML 应用需求……48
3.1.2　XML 的实际应用……52
3.2　XML 的格式……58
3.2.1　XML 文档的组成……58
3.2.2　格式正确的 XML……59
3.2.3　有效的 XML……59
3.3　设计 XML 词汇表……63
3.3.1　XML 词汇表……63
3.3.2　词汇表创建原则……63
3.3.3　词汇表比较……63
3.4　命名空间……65

3.4.1 命名空间 ……65
3.4.2 使用默认命名空间 ……65
3.4.3 使用显式命名空间 ……66
3.4.4 命名空间 URI……68
3.4.5 命名空间 URI 的选择原则……68
3.5 XML 解析器……69
3.5.1 XML 文档对象模型与简单 API ……69
3.5.2 DOM 节点与 XML 的对应 ……71
3.5.3 DOM 节点和相关的节点类型 ……72
3.5.4 支持 DOM 的类……73
3.5.5 从 XML 源中加载 DOM ……74
3.5.6 将 DOM 保存到文档 ……75
3.6 使用 DOM 浏览 XML ……75
3.6.1 Xml 节点 ……75
3.6.2 Xml 节点的属性 ……76
3.6.3 引用单个节点 ……78
3.6.4 解析内存中的 XML 文档对象……79
3.6.5 Xml 有序节点列表 ……81
3.6.6 Xml 无序节点集 ……82
3.7 使用 DOM 创建新节点 ……83
3.7.1 增加节点 ……83
3.7.2 创建元素节点 ……84
3.7.3 为元素节点设置属性……85
3.8 XML 转换概述……87
3.8.1 XSLT……87
3.8.2 XSLT 样式表单的组成部分……88
3.8.3 转换 XML 文档的原因……91
3.8.4 XSLT 结构……91
3.9 XSLT 处理器……93
3.9.1 XSLT 处理器的实现……93
3.9.2 创建 XSLT 处理器对象……94
3.9.3 应用 XSLT 样式表单……95
3.9.4 在 Web 应用程序中显示 XML 数据……98
3.10 扩展 XSLT 样式表单……100
3.10.1 为 XSLT 样式表单传递参数……100
3.10.2 扩展对象 ……103
3.10.3 使用扩展对象的原因……103
3.10.4 将扩展对象传递给样式表单……104
3.10.5 调用扩展对象……105
3.11 参考资源……106
习题……107
第 4 章 文件和数据库访问技术……110
4.1 流……110
4.1.1 流的概念……110
4.1.2 流的基本原理和操作……111
4.1.3 支持缓冲……111
4.1.4 流操作的实现……111
4.1.5 Null 流实例……112
4.2 流的读/写操作……112
4.3 基本文件 I/O ……115
4.3.1 文件流的实现……116
4.3.2 访问文件及其属性信息……117
4.3.3 读取文本的例子……118
4.3.4 写入文本的例子……118
4.3.5 访问目录及其属性信息……119
4.3.6 文件监视系统……120
4.3.7 独立存储……123
4.4 序列化……123
4.4.1 序列化在应用程序中的应用……124
4.4.2 序列化属性……126
4.4.3 对象图……126
4.4.4 序列化过程……127
4.4.5 序列化示例……127
4.4.6 反序列化示例……128
4.4.7 自定义序列化……129
4.4.8 自定义反序列化……130
4.4.9 自定义序列化示例……130
4.4.10 安全问题……132
4.5 数据库访问基础……133
4.5.1 数据库访问技术的需求……133
4.5.2 ODBC 与 OLE DB 简介 ……134
4.5.3 JDBC 简介……135
4.5.4 ADO 简介……136
4.5.5 ADO.NET 概述 ……137
4.6 连接数据源……141
4.6.1 连接数据库的两种方式……141

4.6.2　选择数据提供程序 ……142
4.7　数据访问方式 ……143
4.7.1　选择数据访问方式 ……143
4.7.2　使用数据集访问数据 ……144
4.7.3　使用数据阅读器访问数据 ……148
4.7.4　使用数据集和数据阅读器访问数据的过程 ……151
4.8　使用存储过程 ……151
4.8.1　调用存储过程 ……152
4.8.2　传递参数 ……153
4.8.3　调用行为存储过程 ……154
4.9　参考资源 ……155
习题 ……156
第 5 章　线程和异步编程 ……158
5.1　多任务处理 ……158
5.2　线程介绍 ……159
5.2.1　线程和进程 ……159
5.2.2　.NET Framework 线程体系结构 ……162
5.3　委托 ……164
5.3.1　委托和线程 ……164
5.3.2　使用委托 ……164
5.3.3　多路广播委托 ……167
5.4　使用线程 ……171
5.4.1　启动线程 ……171
5.4.2　操作线程的属性和参数 ……172
5.4.3　管理线程 ……174
5.4.4　线程本地存储区 ……179
5.4.5　中断和终止线程 ……180
5.5　线程安全 ……181
5.5.1　线程安全概述 ……182
5.5.2　同步上下文 ……183
5.5.3　同步代码区域 ……185
5.5.4　手动同步 ……189
5.5.5　线程安全的实现 ……193
5.6　线程的相关技术 ……193
5.6.1　定时器 ……194
5.6.2　线程池 ……195
5.6.3　使用多线程与 Windows 窗体控件 ……197
5.6.4　使用线程的最佳实践 ……199
5.7　异步编程 ……200
5.7.1　异步编程的实现方法 ……200
5.7.2　异步编程和多线程 ……201
5.7.3　异步编程设计模式 ……201
5.7.4　异步文件流读取示例 ……203
5.7.5　异步委托 ……204
5.8　参考资源 ……208
习题 ……209
第 6 章　分布式组件技术 ……212
6.1　分布式组件技术概述 ……212
6.1.1　分布式组件技术的需求 ……213
6.1.2　分布式组件模型和应用 ……214
6.2　远程处理 ……218
6.2.1　远程处理概述 ……219
6.2.2　信道和格式化程序 ……219
6.2.3　激活和代理 ……221
6.2.4　基于租约的生存期 ……223
6.2.5　对象封送处理 ……224
6.2.6　服务器端 ……226
6.2.7　客户端 ……227
6.2.8　客户端编译技术 ……228
6.3　远程处理配置文件 ……228
6.4　分布式组件应用示例 ……230
6.4.1　创建本地应用程序 ……231
6.4.2　添加分布处理能力 ……236
6.5　参考资源 ……242
习题 ……243
第 7 章　XML Web Service ……244
7.1　XML Web Service 概述 ……244
7.1.1　应用程序开发面临的挑战 ……244
7.1.2　XML Web Service 应对挑战 ……245
7.1.3　什么是 XML Web Service ……245
7.1.4　XML Web Service 的实际应用 ……246
7.1.5　XML Web Service 的工作原理 ……247
7.2　XML Web Service 开发 ……248
7.2.1　创建 XML Web Service ……249
7.2.2　创建 XML Web Service 方法 ……250

7.2.3 调试 Web Service ……254
7.2.4 XML Web Service 注册和发现……262
7.2.5 访问 XML Web Service ……263
7.3 参考资源 ……264
习题 ……264
第 8 章 设计模式与构建 ……265
8.1 设计模式概述 ……265
8.1.1 什么是设计模式 ……265
8.1.2 为什么要使用设计模式……265
8.1.3 设计模式与企业解决方案……266
8.2 组织模式 ……268
8.2.1 模式的嵌套使用 ……268
8.2.2 模式群集 ……268
8.2.3 模式的抽象级别 ……269
8.2.4 视点 ……270
8.3 Web 表示模式 ……271
8.3.1 Web 表示模式概述 ……272
8.3.2 Web 表示模式与传统设计模式 ……274
8.4 部署模式 ……288
8.4.1 部署模式概述 ……288
8.4.2 部署模式与传统设计模式……290
8.5 分布式系统模式 ……291
8.5.1 分布式系统模式概述……291
8.5.2 分布式系统模式与传统设计模式……296
8.6 服务模式 ……296
8.6.1 服务模式概述 ……297
8.6.2 服务模式与传统设计模式……299
8.7 性能和可靠性模式 ……300
8.7.1 性能和可靠模式概述……300
8.7.2 性能和可靠性模式与传统设计模式 ……301
8.8 参考资源 ……302
习题 ……302
第 9 章 面向服务的架构（SOA） ……303
9.1 概述 ……303
9.1.1 什么是服务 ……304
9.1.2 什么是 SOA ……305
9.2 SOA 模型 ……307
9.2.1 SOA 基础……307
9.2.2 面临的问题……308
9.2.3 服务系统架构……310
9.2.4 服务模型……313
9.3 服务设计概念……314
9.3.1 服务是一项长期投资……314
9.3.2 企业级应用程序与 SOA……315
9.4 创建面向服务的解决方案……315
9.4.1 面向服务的分析……315
9.4.2 服务的设计……318
9.4.3 服务的管理……319
9.4.4 面向服务解决方案中的设计模式……319
9.5 总结……320
9.6 参考资源……321
习题……321
第 10 章 开发智能设备应用程序 ……322
10.1 引言……322
10.2 微软移动开发平台概述……322
10.2.1 客户端……323
10.2.2 工具和服务器……324
10.3 使用 ASP.NET Mobile Control 开发针对智能设备的 Web 应用 ……325
10.3.1 背景……325
10.3.2 ASP.NET Mobile Control 的工作原理 ……326
10.3.3 WAP 设备访问时的工作流程……327
10.3.4 HTML 设备访问时的工作流程……328
10.3.5 创建移动备忘录应用程序……328
10.3.6 测试……333
10.3.7 小结……333
10.4 开发基于.NET 精简版的智能客户端程序 ……333
10.4.1 .NET Framework 精简版概述……333
10.4.2 用户界面设计……334
10.4.3 远程数据交互……336
10.4.4 测试……342
10.4.5 小结……343
10.5 智能设备应用程序的发布和安全……343

10.5.1　智能设备程序的发布……343
10.5.2　智能设备的安全……348
10.5.3　小结……349
10.6　Java 移动开发平台概述……349
10.6.1　J2ME 概述……349
10.6.2　J2ME 体系结构……350
10.6.3　J2ME 程序的开发……351
10.6.4　小结……355
10.7　总结……355
10.8　参考资源……355
习题……355
词汇表……357

第 1 章　高级开发概述

随着计算机技术以及社会信息化的发展，计算机软件系统越来越多地应用于社会的各行各业中，涉及面也越来越广。其中有些是与人们日常生活相关的，例如在商场里购物，可以使用导购系统来查看商铺位置、商品信息等；其他一些则是与企业管理或日常运营相关的，例如使用计算机来管理货物运输及仓储，或是管理工资信息等。这些系统如果只需要使用本机上的资源，那么可以是运行在单机上的桌面应用系统；如果还需要访问其他计算机上的资源，那就可以是客户端/服务器结构的应用系统。

互联网以及高速宽带网的普及使得基于浏览器/服务器的应用系统成为近年来的一种趋势。这种趋势对应用系统的开发提出了更高的要求。最基本直观的变化就是服务器不仅要支持桌面应用程序客户端，还要支持 Web 浏览器客户端。另外，智能手机设备的出现使客户端的种类更为多样化。这样，无论是服务器端还是客户端的应用开发，其复杂性都大大增加了。

这种复杂性的增加就要求开发人员不能仅仅懂得编程语言的语法，而且要熟悉这些应用开发中所需要使用的技术。那么一个应用系统的开发到底会包含哪些技术？这里通过对一个图书馆应用系统的简单剖析来大致做一个说明。该系统结构如图 1-1 所示。

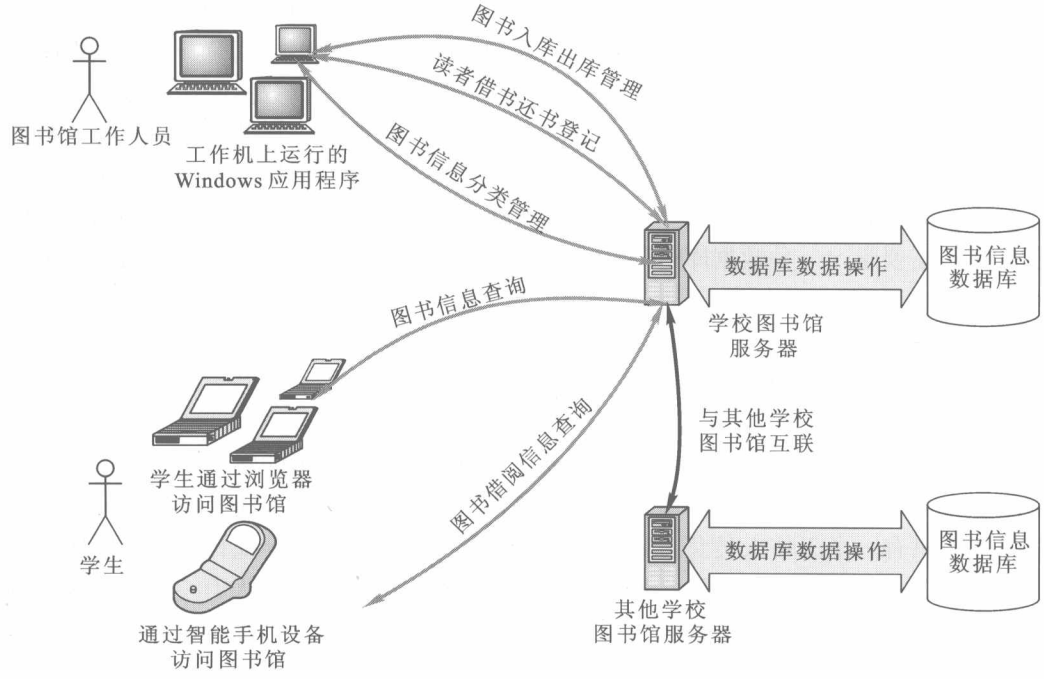

图 1-1　图书馆系统示意图

可以看到，这不算是一个很复杂的应用系统，但“麻雀虽小，五脏俱全”。这个系统中其实包含了本书后面会讲到的各种开发技术。从图 1-1 中可以看到，这个系统分成这样几个部分：

- 数据库
- 服务器
- Windows 应用程序（桌面应用程序）
- Web 应用程序
- 智能手机设备应用程序

下面列出的是这个系统的基本需求：

- 图书馆工作人员在使用系统时需要使用用户名和密码进行登录
- 学生在使用系统时需要使用学号和密码进行登录
- 对图书馆中所有图书的信息（包括书名、书号、借阅状态等）进行存储和管理
- 购买新书和处理旧书时，需要更新图书的信息
- 图书归还逾期或图书损毁时自动计算罚款金额和产生罚款通知
- 学生借阅和归还书籍时，工作人员需要更新图书和读者的借阅信息
- 学生可以在寝室或家中通过自己的计算机查询学校图书信息以及自己的图书借阅信息
- 学生可以在寝室或家中通过自己的计算机查询其他学校的图书信息
- 学生可以通过智能手机设备访问图书馆及查询信息

1.1 数据存储与访问

图书馆系统最基本的功能就是对图书信息的存储和管理。一般少量的数据信息，可以选择存放在文件系统中。但是如果数据的数量大、复杂度高，而且数据之间有一定关联性时，则会选择存放在数据库中。这样系统在需要访问这些数据时，就会用到数据库访问技术。现在流行的数据库种类很多，比较著名的有 SQL Server、Oracle、DB2 等。这些数据库本身的数据存放组织方式都不尽相同。比较优秀的数据库访问技术必须要能够屏蔽这些不同点，用一种比较统一的方式去访问不同数据库中的数据。这样就能减少代码量，降低数据访问代码的复杂性。这样开发人员只需要学习一种数据库访问技术，就可以灵活地操作各类数据库，而无须了解这些数据库的实现细节。

1.2 多线程编程技术

实现了数据的存储和访问以后，就会引出下一个需求。学校里的学生和图书馆工作人员都需要使用这个图书馆系统，那自然需要服务器能够同时并行响应请求和处理多个任务。这就像图书馆里如果只有一个工作人员负责处理借书和还书，那就一定会排很长的队；但如果可以有几个甚至十几个工作人员一起处理，那速度就会大大提高，不需要排队了。这其实就是所谓的多线程技术。服务器必须提供这种多线程处理的能力，以便应付多个任务和请求。对图书馆系统来说服务器端需要处理两类任务请求，一类是来自于工作人员的桌面应用程序，另一类是学生通过浏览器（位于另外的计算机或智能手机设备）发出的请求。一般来说，后一类请求的多

线程处理是通过 Web 服务器来实现的，不需要开发人员编写额外的代码，而前一类请求的多线程处理就要求开发人员编写多线程处理代码来实现了。

不仅服务器端需要使用多线程技术进行处理，工作人员使用的 Windows 客户端也需要使用多线程技术。举一种情况来看，图书馆里的图书信息成千上万，对这些信息的查询处理需要一定的时间，再加上网络传输的延迟，所花费的时间就相当可观了。如果不采用多线程处理，当系统在查询数据的时候，就无法响应用户的其他操作了，诸如移动窗口、点击按钮等。这会给使用带来很大的不便。因此，客户端的开发也需要使用多线程技术。

1.3 组件化开发技术

工作人员和学生在登录系统时，需要输入登录信息。其中密码的验证是通过服务器来进行的。如果把密码不经加密直接在网络中进行传输，如果被截取就会对系统的安全造成极大损害。因此需要对密码进行加密，而加密的算法是需要保密的，不应该让所有的开发人员了解。这样，就需要把加密和解密的代码打包为二进制文件，只提供加密和解密的函数接口。这样其他开发人员即使不了解加密和解密算法，通过这个接口也能实现对密码的验证。并且这个加密和解密的功能可以同时为工作人员和学生使用的应用程序使用，而不需要重复开发。事实上，这种打包的方法就是组件化技术。这样做还有一个优点，那就是即便以后要改进加密和解密的算法，也只需修改加密和解密的那部分，而不会影响其他的代码，也就为以后的维护提供了方便。在本书下面的内容中，还可以了解到组件化开发的其他特点。

1.4 XML

学生登录系统以后，可能还需要查阅其他学校图书馆中的内容。但是本校和其他学校的图书管理系统可能不同，甚至图书信息的存储格式也不完全相同，但其实这些信息中包含的数据却很相近，例如书名、书号、借阅状态等，不同之处仅仅在于数据需要以不同的格式呈现。为了减少在两种不同格式之间保持数据的对应和一致所花费的时间，就需要一种统一的方式来描述这样的数据。这个问题可以用一种新的数据描述方法——XML（eXtensible Markup Language，可扩展标记语言）——来解决。XML 语言对数据的描述比较统一和直观，像这样为了在两个不同的系统之间传递数据，一般都会使用 XML 语言编写 XML 文件来描述记录数据。在一些常用的编程语言中都提供了对 XML 语言的支持和对 XML 文件的读/写功能，所不同的只是使用的灵活性和直观性。

1.5 远程调用与 XML Web Service

当然，要获得这些 XML 语言记录的数据信息，必须以某种方式通知其他学校的服务器所需要查询的条件，再根据这些条件取得相应的信息。以技术的语言来说，就是需要调用其他学校服务器上的功能，输入查询条件，输出查询结果。这里就引出另一个问题，如何去调用其他系统中的功能。在这里有两种方式可以选择，它们都可以直接使用其他系统中的功能。

一是远程调用。传统的远程调用有很多种具体的实现方法，这些方法的技术细节、传输格式各不相同，很难以一种统一的方法进行使用，而且编程比较困难。而新的远程调用技术在这方面有所改进，使用更为简单，具体内容会在下文中进行介绍。但使用这种新的技术要求不同的系统在开发时就使用同一种语言进行开发，这带来了一定的限制。

另一种方法就是 XML Web Service，也称为 Web 服务。如果两个系统所运行的平台、开发所使用的语言都不尽相同，要在它们之间实现相互通信和协作，比较好的一种方式就是使用 XML Web Service。任何一个系统只要根据一套定义的标准，都可以把本系统内的某些功能以 Web 服务的形式提供出来。其他系统通过获得这个服务的描述信息，从而了解这个服务的功能以及使用方式，进一步就能够使用这个服务来完成所需的功能。而且服务所提供的信息是以统一的 XML 语言进行描述的，这就保证了通信的一致性。

1.6 文件 I/O 流

另外，根据需求，学生归还图书时，如果出现逾期或图书损毁现象，就需要生成一个罚款清单。例如，可按照下面的形式生成：

```
学生姓名：XXX
学号：XXXXXXX
所借书籍：XXXX
逾期天数：X 天
罚款金额：X.XX 元
```

也就是说，需要开发人员编程自动生成一个文本文件，文件内容一部分是事先确定的，另一部分是临时获得或生成的。这里，就需要使用文件输入/输出流（I/O stream）技术来操作文件。

1.7 智能移动设备

拥有智能手机设备的学生可以直接通过这些设备连接图书馆系统查询相关图书信息。要开发这样的系统，就需要了解智能手机开发技术。这种开发技术发展至今，已经与通常应用系统开发技术区别不大了。而且，通过智能手机无线上网也正成为一种趋势。因此，这种技术也成为现在开发人员研究的方向之一。

1.8 虚拟执行环境

前面所做的这些分析和引出的开发技术基本上已经能够满足开发一个图书馆应用系统的要求。但是在这些技术的背后还有一个重要问题需要阐述，就是应用系统的执行环境，即这个系统运行在一个什么平台上。

一种情况下，应用系统直接在操作系统中运行，直接访问内存和系统资源。这种方式是比较传统和直接的。但也带来了一系列的问题，诸如安全性问题、内存泄漏问题等。这些问题对

于现在的应用系统来说，都是至关重要的。有一些计算机病毒，如红色代码、冲击波等，都是利用直接访问内存和系统资源，缓冲区溢出而导致的软件安全漏洞来破坏应用系统的。有过传统语言编程经验的开发人员，都会为内存指针的管理而大费周章。这些都是直接访问内存和系统资源带来的问题。

另一种情况下，应用系统运行在一个虚拟执行环境中。虚拟执行环境就其本质而言是一个软件平台，其功能是执行其他应用程序。由于在硬件和其他应用程序之间存在虚拟执行环境这一软件平台，应用程序无法直接访问硬件设备，规避了上述直接访问硬件而导致的漏洞。而且正由于这个原因，它可以提供一些硬件不能实现的功能，例如自动管理内存分配、对代码的安全性做更严格的检查等。而且，现代的虚拟执行环境都带有一套完整的开发库，提供了丰富的功能，也简化了开发者的学习负担，提高了开发效率。

1.9 开发技术总结

以上对这样一个图书馆系统的开发所需的技术进行了分析，可以总结如下：

- 组件技术
- 虚拟执行环境
- 文件 I/O 流
- 多线程技术
- XML 语言
- 数据库访问技术
- 远程调用技术
- XML Web Service
- 智能手机开发技术

1.10 架构和设计模式

分析完这些技术以后，再来看看另一个问题，就是这个系统的结构，也就是常说的系统架构。前面谈到的技术，其实都是对这个架构的具体实现。那么架构在当今的应用开发中到底扮演着什么样的角色呢？

第一，架构是整个应用系统的蓝图。架构对于开发应用系统的意义就相当于房屋结构设计图对于建造房屋的意义。架构的设计比技术的使用要高一个层次。换句话说，决定了采用什么样的架构，才能继续选定要实现这样的架构所需使用的技术。当然这不是一个绝对的过程，因为在考虑架构设计的时候，必须清晰地了解这样的架构在目前技术条件及其他限制条件下的可行性。这就好比不能设计一间高 100 km 的房子一样。

第二，一个好的架构能够从各方面对应用系统的开发产生积极影响。这些方面通常包括开发效率、可靠性、安全性、可维护性、可管理性、可扩展性和性能等。这里以图书馆系统为背景举几个例子来说明架构如何对这些方面产生影响。

- 安全性。如果在架构设计中由客户端来完成密码验证，而不是由服务器端来验证，那自

然就需要在客户端保存所有的用户登录信息。这样的信息就很容易被别人窃取而造成安全性问题

- 可维护性。架构设计中，服务器端有一层数据访问层，它负责所有与数据库的交互。如果数据库的结构发生变化，那么就只要在该层修改即可。但如果访问数据库的代码不是封装在一层中，而是散布在应用系统的各处，那么就需要寻找各处进行修改，不仅费时，而且容易出错
- 可扩展性。一台计算机能够服务的客户数量是有限的，当系统被越来越多的用户使用时，应用系统必须能够扩展到多台计算机上，利用多台计算机的处理能力来服务更多的用户。很明显，由多台计算机组成的系统要比单机复杂，尤其需要处理好的是多台计算机之间的分工协作，使其不会相互干扰。在设计系统的架构时就需要考虑这些问题

综上所述，架构设计对于应用系统开发的重要性十分明显。在下文中会对目前比较先进的架构设计方式进行介绍。

确定了应用系统的架构以后，必须要建立起架构与实际的代码之间的联系。在现今的面向对象编程技术的大环境下，要解决的问题也就是要设计哪些类来实现系统的架构，这些类之间是怎么样的关系。这一步对系统开发也是十分关键的。因为这些类设计的优劣会直接影响代码结构、重用能力、维护效率等多方面。这里要解决的其实是设计的方法论问题，或称为设计模式。本书中会对当前比较流行和占主导地位的设计模式进行介绍，以便开发人员了解开发过程中这重要的一环。

这里虽然是对一个图书馆系统的分析，但是通过这些分析可以看到，其中的一些通用场景，例如不同系统的互联、数据库的访问、多线程技术等，对于大部分日常应用系统和企业应用系统都是适用的。因此，这些技术对于现今的应用系统开发都是必不可少的。每一种技术都有其应用领域并解决相应的需求。本书将会在以后的章节里对这些技术进行更深入的介绍。

第2章　组件化开发

在基于组件的开发理论指导下，系统开发是重用已有组件、开发新组件及整合所有组件的综合过程。本章将会从组件化开发技术在软件系统开发中的作用入手，介绍组件的编写、调用以及基于组件的应用程序部署方案。

学完本章后，将能够：

- 掌握组件化开发方法
- 掌握组件的创建和调用
- 掌握应用程序部署方案的设计

2.1　组件化开发概述

在现实生活中有很多工业产品都遵循一定的制造标准，因而能够相互配合使用。例如，可更换笔芯的铅笔外形各异、颜色不同，但是都可以使用同一尺寸的铅芯；组成一台 PC 机的部件可能来自数十家甚至更多的厂商，这些部件却可以轻易地组合起来。原因是这些产品在设计制作过程中遵循了一定的标准。

（1）标准化与组件技术

制造业发展到一定程度必然需要各种标准的出现，即使小到螺丝、螺母的配合也有相应的标准。标准化可以简化设计制造过程，降低制造的成本，使不同厂家的产品也能很好地配合使用。

软件业也是一样，软件模块也应像工业产品一样，通过遵循某些标准或协议能够灵活组合替换，从而加快开发进度，提高效率。组件技术和组件化开发技术由此产生。

所谓组件是指可用于重用、发布和部署的二进制代码单元。与制造业中的标准类似，组件技术包含了不同软件交互所需的规范。只要遵循规范，开发出的组件就可以相互兼容，多个组件可以结合起来发挥作用。例如开发应用程序时，如果某个部分或是难度较高或是实现繁琐，那么可以在市场上购买现成的组件直接利用。正如汽车生产厂本身决不会自行生产每个铆钉，也不一定需要具有设计制造发动机的能力，这些部件都可以从市场上买到，装配起来即可以得到最终产品——汽车。

组件化开发（或者称为基于组件的开发）是目前应用软件开发中常用的开发方法。组件化开发借助于现代软件技术中的组件技术，利用了组件本身就是软件开发、部署、重用的基本模块这一特点，使开发过程更加高效经济，部署方式多样灵活，程序维护比以往更简单。

（2）接口与实现分离

组件化开发中一条基本原则是接口与实现分离，这很容易用组装计算机的例子来说明。假

设有一台计算机，因为每天要存储许多从互联网下载的文件资料，磁盘空间不足了，这时可以很容易地从市场上买到标准的 IDE 插口硬盘，将硬盘连接到计算机上，解决空间不足的问题。仔细考虑购买硬盘的过程，用户会选择不同品牌的硬盘，即使是相同品牌的不同型号产品，在容量、转速和缓存大小方面也各有不同。这些因素会影响用户的购买选择，但是用户根本不必担心买回去的硬盘会无法安装到计算机上去。只要是 IDE 接口的硬盘，总是可以连接到 PC 机上。在这里关键的标准是硬盘与计算机的连接方式，只要两者遵循同一个标准，无论容量、转速和缓存大小如何不同，硬盘一定可以使用。硬件如此，软件也一样，在编写组件时只要遵循相同的接口，组件之间就可以配合使用，而实现的方式会依具体情况而定。组件化开发中接口和实现相分离的原则是组件和开发技术灵活性的来源。

通常用图 2-1 中的图示来表示组件，示意图的画法也表明了接口是组件设计实现时需要关注的部分，组件通过接口与外部应用程序交互。而在组件内部实现接口的方式是外部代码不可见的，因此用一个封闭的方框表示。

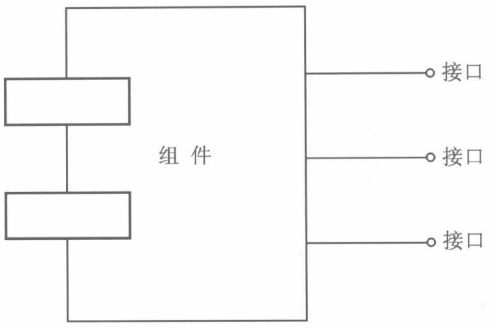

图 2-1　接口和组件示意图

当应用程序使用基于组件的方式开发时，则可以利用组件基于接口的优点，灵活组合。一方面应用程序可以由自行开发的组件组成，或通过购买获得；另一方面应用程序也可以通过同样的接口操作功能不同的组件。如图 2-2 所示。

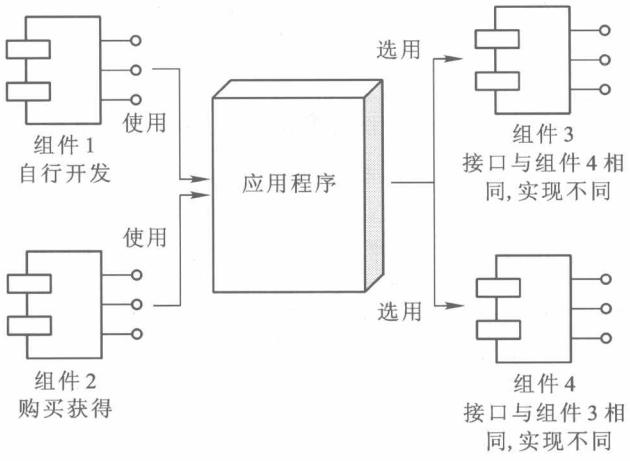

图 2-2　组件化开发的灵活性

（3）组件技术解决的问题

传统技术中代码的重用和部署比较繁琐。静态库（static library）不够灵活，必须在编译时和主程序链接在一起，结果是程序文件变得更大。如果静态库中的代码因为某些原因需要修改（例如修正缺陷），则必然导致对主程序重新编译。动态链接库（dynamic link library，DLL）相对灵活很多，但是动态链接库在更新和多版本共存等方面还是有比较明显的缺陷。最典型的即