

图书基本信息

书名：<<VISUAL BASIC6高级编程策略与范例--错误处理编码与分层技术>>

13位ISBN编号：9787505359345

10位ISBN编号：7505359347

出版时间：2000-05-01

出版时间：电子工业出版社

作者：(美)Tyson Gill

页数：239

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

## 内容概要

Visual Basic往往被人们认为是一种易学、易用，但不稳定、不可靠的编程语言，而本书的作者根据自己的多年编程经验，提出了灵巧编码三角和安全编程框架的理论，可大大改善这一局面。本书分为错误处理编码和分层应用程序开发技术两大部分。第一部分通过介绍可重用性、标准化和错误处理编码这三部分构成的灵巧编码三角，来达到实现有效错误处理编码和预防错误的目的，第二部分介绍的分层的开发策略，大大提高了编程的效率和质量，降低了出错的可能性。本书提出的理论来源于实践，实用且有效。不仅对Visual Basic的初学者、分析人员、管理人员，而且对使用各种编程语言的编程高手来说，都是其提高编码质量，改善开发策略的良师益友。

## 书籍目录

第1章 软件开发的任务1.1 草拟任务1.2 保留公司的知识1.3 创建过程的标准化1.4 错误处理编码1.5 灵巧编码1.6 鉴别这种可能性1.7 实现这种可能性1.8 灵巧编码三角1.9 实现任务的障碍第2章 理解任务的障碍2.1 Visual Basic错误处理编码2.2 为什么好的错误处理编码很少能实现2.2.1 样本代码将注意力放在了功能上2.2.2 错误处理编码不具魅力2.2.3 错误处理编码难以掌握2.2.4 错误处理编码难以实现2.2.5 错误处理编码被看作是辅助性的2.2.6 错误处理编码被认为是不言而喻的2.2.7 错误处理编码需要大量的代码2.2.8 错误处理编码不可能正确地分别实现2.2.9 错误处理编码在最终产品中是不可见的2.2.10 错误处理编码是首先被节省的部分2.2.11 如果确实节省了错误处理编码部分会怎样？2.2.12 即使后来出现了错误，当它们出现时总是可以被修复2.2.13 管理设立了障碍2.2.14 认为代码是一次性的2.3 “我们将适应”2.4 获得好的错误处理编码2.5 错误处理编码的障碍2.6 评估错误处理编码2.7 代码标准化的障碍2.8 代码重用的障碍2.9 消除障碍第3章 有效错误处理编码的实现3.1 提高我们的期望3.2 尽早处理错误3.3 随时进行错误处理编码3.4 预料错误3.5 预防错误3.6 处理错误3.7 捕获错误3.8 报告错误3.9 避免假设3.9.1 我将再也不需要使用这些代码3.9.2 我是唯一接触到这些代码的人3.9.3 我仅仅为一个特殊的情况而设计它3.9.4 一般编码假设3.10 为重用设计函数3.11 重用错误处理编码3.12 系统化错误处理编码第4章 显式编码4.1 显式变量的用法4.1.1 永远使用Option Explicit4.1.2 显式地指出变量类型4.1.3 避免DefType语句4.1.4 使用指定的数据类型4.1.5 初始化所有变量4.1.6 一行使用一个变量4.1.7 使用TypeName、VarType及TypeOf4.1.8 使用枚举4.2 参数4.2.1 永远使用ByVal或ByRef4.2.2 显式地指明参数的类型4.2.3 为可选参数设置显式缺省值4.2.4 验证所有参数4.2.5 使用命名的参数4.3 数组4.3.1 永远不要假设数组的下界4.3.2 不要硬编码数组的界4.3.3 避免使用Option Base4.4 编码建议4.4.1 永远包括Else4.4.2 避免使用缺省属性4.4.3 避免在表达式中混合使用数据类型4.4.4 使用常量 (Constant) 4.4.5 避免使用操作符优先级4.4.6 检查字符串长度4.4.7 关闭所有打开的对象4.4.8 设置对象为Nothing4.4.9 永远显式地关闭错误捕获4.4.10 永远不要对外部世界做任何假设4.4.11 不要剪切并粘贴4.4.12 正确使用 + 和 & 4.4.13 伪代码4.4.14 在运行时设置属性4.5 基本原则：显式编码第5章 错误处理编码机制5.1 错误处理编码并非是一种假设5.2 Visual Basic错误处理5.3 没有错误处理5.4 错误处理器5.5 On Error Resume Next5.6 错误抑制5.7 On Error Goto5.8 恢复程序执行5.8.1 Resume5.8.2 Resume Next5.8.3 Resume line5.9 多个错误处理器5.10 检查错误5.11 检查Err.Number5.12 处理错误5.13 清除Error对象5.14 关闭错误处理器5.15 错误处理的范围5.16 错误冒泡5.17 错误中的错误5.18 修改错误处理器5.19 错误捕获5.20 错误捕获处理器5.21 内联处理错误5.22 引发错误5.23 错误捕获块与内联错误处理器5.24 何时使用错误捕获块5.25 何时使用内联错误处理5.26 完全避免错误处理第6章 错误预防6.1 错误类型6.1.1 程序错误6.3.1 长远考虑6.3.2 为他人编写6.3.3 防御性编码6.3.4 攻击性编码6.3.5 避免错误抑制6.3.6 雅洁性预防技术6.3.7 同一个地方不能跌倒两次6.3.8 不要修复同一个错误两次6.3.9 重用6.3.10 标准化6.3.11 包装系统函数6.3.12 不要对预防使用错误捕获6.4 预防用户错误6.4.1 程序设计的三条基本原则6.4.2 使用显式方式6.4.3 精炼所做的设计6.4.4 使用户界面清晰6.4.5 使消息清晰6.4.6 过滤用户输入6.4.7 确认用户输入6.4.8 使用控件数组6.4.9 选择正确的控件6.4.10 包装控件6.5 窗体预防习惯第7章 安全编码框架7.1 可重用性例程7.2 安全过程7.3 安全函数7.3.1 忽略错误7.3.2 报告错误7.3.3 传回错误7.3.4 返回一个新的错误7.3.5 添加检查跟踪7.3.6 处理错误7.4 安全错误消息7.5 防御函数7.6 防御于例程7.7 安全类7.8 重用SPF程序7.9 自完备程序7.10 代码块7.11 命名约定7.12 参数7.13 限定范围7.14 计数器变量7.15 修订版编号7.16 可重用性文档7.16.1 过程头7.16.2 注释头示例7.16.3 版本历史注释7.17 清除7.18 使用SPF7.19 实现标准第8章 SPF实例8.1 通用结构8.2 安全错误应用8.2.1 创建安全错误消息8.2.2 在安全错误消息中统计错误8.2.3 分析安全错误消息8.2.4 报告安全错误消息8.3 数组处理8.3.1 获得数组下界8.3.2 获得数组的两个界8.3.3 获得数组计数8.4 类型转换和数据验证8.4.1 转换字符串8.4.2 转换日期8.4.3 转换数字8.4.4 验证数字8.5 字符串处理8.5.1 安全Len包装程序8.5.2 SSN格式8.6 窗体和控件8.6.1 判断窗体是否载入8.6.2 卸载所有窗体8.6.3 设置焦点8.6.4 确定调整大

小安全8.6.5 拷贝List控件8.7 数据库例程8.7.1 格式化SQL字符串8.7.2 检查游标的位置8.7.3 编辑域8.8 使安全过程正常工作第9章 团体策略9.1 灵巧编码团队9.2 合作的竞争9.3 开发自己的标准9.4 创建安全过程9.5 动机催化9.6 认证可重用9.7 共享被认证的过程9.8 使用超级库9.9 奖励永久的贡献9.10 通过认证检查代码9.11 适应性开发9.12 消除非技术的障碍9.13 还差一步！

第10章 编程体系10.1 程序失败的方式10.1.1 结束失败10.1.2 执行失败10.1.3 维护失败10.2 串珠程序10.3 可维护性10.4 维护噩梦10.5 隐式事务逻辑10.6 体系结构的量度10.7 叶和层10.8 普遍的分层结构10.8.1 用户层10.8.2 事务层10.8.3 数据层10.8.4 用户连接层10.8.5 数据库层10.8.6 数据连接层10.9 可重用层10.10 分层的流10.11 分层与绑定10.12 分层与类10.13 分层与纽10.14 层打包10.15 配置层10.16 分层的优点10.17 学习分层实例10.18 数据集10.19 不受技术变化的影响10.20 实现分层的应用程序第11章 设计分层的应用程序11.1 数据库11.2 规划数据层11.3 映射控件11.4 创建层11.5 伪编码用户层11.5.1 Form Load11.5.2 Stuents Click11.5.3 Customers Click11.5.4 编辑域11.5.5 保存变化11.5.6 添加和删除11.6 伪编码事务层11.6.1 显示数据集11.6.2 显示被计算的域11.6.3 改变数据集中的数据11.7 数据层的方法11.8 用户连接层的方法11.9 数据连接层的方法11.10 分层应用程序的优点11.11 创建一个安全的分层库11.12 使用层包装控件11.13 实际应用第12章 完成任务12.1 获得全方位视图12.2 评价成功12.3 进行下一步12.4 继续前进！

附录A 命名约定附录B SPF检查表附录C 认证评价单

版权说明

本站所提供下载的PDF图书仅提供预览和简介, 请支持正版图书。

更多资源请访问:<http://www.tushu007.com>