

<<精通C#编程>>

图书基本信息

书名：<<精通C#编程>>

13位ISBN编号：9787121143137

10位ISBN编号：7121143135

出版时间：2011-10

出版时间：电子工业出版社

作者：郑阿奇 主编，袁永福，张小勇 编著

页数：664

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

本书以C#编程为基础，系统介绍C#高级编程技术，主要包括图形开发、XML开发及.NET框架的反射等内容。

C#图形开发主要包括图形开发基础、桌面图形开发和Web图形开发。

XML技术开发包括XML基础、安全获取数据、XML/XSLT开发和基于XSLT的代码生成器等。

.NET框架的反射技术包括使用反射和特性构造ORM框架、基于反射和动态编译的快速ORM框架。

其他方面包括高性能ASP.NET树状列表控件、验证码技术、可快速绑定数据源程序框架和基于动态编译的VB.NET脚本引擎，以及C#服务端程序的编程和文档对象模型等。

本书所有实例在VS.NET

2010专业版环境下开发，所有源代码文件、工程文件和同步教学课件包含在本书配套的光盘中，书中所有的源代码均可在VS.NET

2005、2008版环境下运行。

书籍目录

第1章 C#图形开发基础

- 1.1 Windows图形
 - 1.1.1 图形子系统基本原理
 - 1.1.2 图形设备上下文
 - 1.1.3 用户界面事件
 - 1.1.4 图形用户界面闪烁
- 1.2 C#图形开发基础
 - 1.2.1 GDI+概述
 - 1.2.2 矢量图和位图
 - 1.2.3 图形坐标系统
 - 1.2.4 图形开发基本原理
 - 1.2.5 C#图形开发基础
- 1.3 C#画图初步
 - 1.3.1 画布
 - 1.3.2 画笔
 - 1.3.3 画刷
 - 1.3.4 颜色
 - 1.3.5 绘制直线
 - 1.3.6 绘制矩形
 - 1.3.7 绘制椭圆
 - 1.3.8 绘制圆弧
 - 1.3.9 绘制多边形

第2章 自定义图形交互按钮

- 2.1 功能需求
- 2.2 软件设计
- 2.3 软件开发
 - 2.3.1 新建Visual C# 的Windows应用程序工程
 - 2.3.2 新增按钮控件
 - 2.3.3 定义按钮的属性
 - 2.3.4 绘制控件用户界面
 - 2.3.5 响应鼠标事件, 实现动态效果
 - 2.3.6 触发Click事件
 - 2.3.7 测试控件
- 2.4 完成开发

小结

第3章 自定义数据网格控件

- 3.1 功能需求
- 3.2 软件设计
 - 3.2.1 文档对象模型
 - 3.2.2 视图控件
- 3.3 软件开发
 - 3.3.1 建立表格文档对象模型
 - 3.3.2 创建视图控件类型
 - 3.3.3 加载数据
 - 3.3.4 内容排版

<<精通C#编程>>

- 3.3.5 绘制用户界面
- 3.3.6 处理鼠标事件
- 3.3.7 复制数据
- 3.3.8 系统预定义颜色
- 3.3.9 折射效应
- 3.4 测试控件

小结

第4章 鼠标签名

- 4.1 功能需求
- 4.2 软件设计
 - 4.2.1 文档对象模型
 - 4.2.2 视图控件
- 4.3 软件开发
 - 4.3.1 PointArrayList 轨迹点坐标数据列表
 - 4.3.2 PenMarkInfo签名信息对象
 - 4.3.3 PenMarkInfoDocument签名信息文档对象
 - 4.3.4 PenMarkControl签名视图控件

小结

第5章 C# Web图形开发基础

- 5.1 Web开发和桌面图形开发比较
- 5.2 C# Web图形开发基本原理

小结

第6章 带超链接的饼图设计

- 6.1 概述
- 6.2 关键技术
 - 6.2.1 map标签
 - 6.2.2 Session对象
 - 6.2.3 GraphicsPath类
 - 6.2.4 Guid结构
 - 6.2.5 Math类
- 6.3 软件设计
 - 6.3.1 文档对象模型
 - 6.3.2 程序结构设计
 - 6.3.3 HTML设计
 - 6.3.4 模拟扇形几何算法
- 6.4 软件开发
 - 6.4.1 新建ASP.NET应用程序项目
 - 6.4.2 饼图项目类
 - 6.4.3 饼图项目列表类
 - 6.4.4 饼图文档类
 - 6.4.5 主页面
 - 6.4.6 临时文件页面
 - 6.4.7 订单信息页面
 - 6.4.8 图片数据服务页面

小结

第7章 XML基础

- 7.1 XML发展历史

<<精通C#编程>>

7.2 XML基本内容

7.2.1 XML介绍

7.2.2 XML衍生标准

7.2.3 国际标准

7.3 微软.NET框架对XML的支持

7.3.1 流式处理模型

7.3.2 DOM处理模型

7.3.3 其他处理模型

7.4 XML对Web开发的意义

7.4.1 XML和HTML

7.4.2 XML和WebService

7.4.3 XML/XSLT提供一种全新的开发模式

小结

第8章 安全获取数据库数据记录

8.1 概述

8.1.1 功能需求

8.1.2 需求分析

8.2 基础知识

8.2.1 XmlDocument类

8.2.2 XmlElement类

8.2.3 XmlTextWriter类

8.2.4 DBNull类

8.2.5 Response

8.2.6 using

8.3 关键技术

8.3.1 设置HTTP输出类型

8.3.2 XmlTextWriter与XmlDocument类输出XML文档

8.4 软件开发

8.4.1 新建ASP.NET应用程序项目

8.4.2 index页面设计

8.4.3 UseXmlTextWriter页面设计

8.4.4 UseXmlDocument页面设计

小结

第9章 XML/XSLT开发

9.1 概述

9.2 基础知识

9.2.1 XmlNode类

9.2.2 XmlNodeList类

9.2.3 StringWriter类

9.2.4 XslCompiledTransform类

9.2.5 < pages > 标记与@Page指令

9.2.6 XPath介绍

9.2.7 XSLT介绍

9.2.8 ML/XSLT在Web开发中的应用

9.3 关键技术

9.4 软件开发

9.4.1 新建ASP.NET应用程序项目

<<精通C#编程>>

- 9.4.2 index页面设计
- 9.4.3 head.xml文件设计
- 9.4.4 table.xml文件设计
- 9.4.5 table2.xml文件设计
- 9.4.6 UseXmlDocument页面设计
- 9.4.7 UseXmlTextWriter页面设计
- 9.4.8 TestXPath页面设计

小结

第10章 基于XSLT的代码生成器

- 10.1 基础知识
 - 10.1.1 序列化
 - 10.1.2 XmlSerializer类
 - 10.1.3 特性 (Attribute)
 - 10.1.4 XmlNamespaceManager类
 - 10.1.5 StringCollection类
 - 10.1.6 CommandBehavior枚举
 - 10.1.7 CollectionBase类
 - 10.1.8 ICloneable接口
 - 10.1.9 嵌套类
- 10.2 软件设计
 - 10.2.1 代码生成器
 - 10.2.2 数据来源
 - 10.2.3 代码生成模板
- 10.3 软件开发
 - 10.3.1 FieldInfo类的设计
 - 10.3.2 TableInfo类的设计
 - 10.3.3 DataBaseInfo类的设计
 - 10.3.4 Default页面设计
 - 10.3.5 XSLT模板

小结

第11章 高性能ASP.NET树状列表控件

- 11.1 树状列表控件
 - 11.1.1 分析问题
 - 11.1.2 解决问题
 - 11.1.3 运行软件
- 11.2 基础知识
 - 11.2.1 HtmlGenericControl类
 - 11.2.2 自定义控件
- 11.3 软件设计
 - 11.3.1 结构设计
 - 11.3.2 目标HTML代码设计
 - 11.3.3 脚本设计
 - 11.3.4 节点XML文档设计
 - 11.3.5 XSLT文档设计
- 11.4 软件开发
 - 11.4.1 SkyTreeNode类设计
 - 11.4.2 SkyTreeNodeList类设计

<<精通C#编程>>

- 11.4.3 SkyTreeViewControl文件设计
- 11.4.4 SkyTreeViewControl.xslt文件设计
- 11.4.5 部署控件
- 11.4.6 Default页面设计
- 11.4.7 TreeViewNodeXml页面设计

小结

第12章 验证码技术

- 12.1 概述
- 12.2 关键技术
 - 12.2.1 Size与SizeF结构
 - 12.2.2 Font类
 - 12.2.3 StringFormat类
 - 12.2.4 StringBuilder类
- 12.3 验证码原理
 - 12.3.1 枚举字典安全攻击
 - 12.3.2 验证码防御
 - 12.3.3 验证码技术概念
- 12.4 软件设计
- 12.5 软件开发
 - 12.5.1 新建ASP.NET应用程序项目
 - 12.5.2 验证码图片类
 - 12.5.3 验证码图片服务页面
 - 12.5.4 登录页面

小结

第13章 使用反射和特性构造ORM框架

- 13.1 ORM背景
- 13.2 基础知识
 - 13.2.1 反射与特性
 - 13.2.2 Object类
 - 13.2.3 Hashtable类
 - 13.2.4 TypeConverter类
 - 13.2.5 TypeDescriptor类
 - 13.2.6 IFormattable接口
 - 13.2.7 PropertyInfo类
 - 13.2.8 BindingFlags枚举
 - 13.2.9 IEnumerable接口
 - 13.2.10 Activator类
- 13.3 软件设计
- 13.4 软件开发
 - 13.4.1 BindTableAttribute类的设计
 - 13.4.2 BindFieldAttribute类的设计
 - 13.4.3 MyORMFramework类的设计
 - 13.4.4 DB_Employees类的设计
 - 13.4.5 Default.aspx页面设计
- 13.5 部署ORM框架

小结

第14章 基于反射和动态编译的快速ORM框架

<<精通C#编程>>

- 14.1 动态编译技术
- 14.2 基础知识
 - 14.2.1 Assembly类
 - 14.2.2 CSharpCodeProvider类
 - 14.2.3 CompilerResults类
 - 14.2.4 CompilerParameters类
 - 14.2.5 StringCollection类
 - 14.2.6 DefaultValueAttribute类
- 14.3 软件设计
- 14.4 软件开发
 - 14.4.1 RecordORMHelper类设计
 - 14.4.2 IndentTextWriter类设计
 - 14.4.3 MyFastORMCodeGenerater类设计
 - 14.4.4 MyFastORMFramework类设计
 - 14.4.5 Default.aspx页面设计
- 14.5 部署快速ORM框架
- 小结
- 第15章 可快速绑定数据源程序框架
 - 15.1 数据源绑定信息
 - 15.2 微软.NET框架的设计时支持
 - 15.2.1 属性列表过滤
 - 15.2.2 属性值只读
 - 15.2.3 属性默认值
 - 15.2.4 属性说明文本
 - 15.2.5 属性类别
 - 15.2.6 使用扩展属性值编辑器
 - 15.2.7 自定义扩展属性值编辑器
 - 15.3 软件设计
 - 15.3.1 数据库处理层
 - 15.3.2 系统字典
 - 15.3.3 数据源模块
 - 15.3.4 数据源事件广播器
 - 15.4 用户界面层设计
 - 15.4.1 设计时支持模块
 - 15.4.2 运行时支持模块
 - 15.5 软件测试
 - 15.5.1 测试用的数据库
 - 15.5.2 窗体
 - 15.5.3 程序移植
 - 小结
- 第16章 基于动态编译的VB.NET脚本引擎
 - 16.1 脚本的原理
 - 16.1.1 VB.NET脚本原理
 - 16.1.2 VB.NET脚本引擎设计
 - 16.2 软件开发
 - 16.2.1 参数控制属性
 - 16.2.2 编译脚本

<<精通C#编程>>

- 16.2.3 调用脚本
- 16.3 Window全局对象
 - 16.3.1 成员属性
 - 16.3.2 延时调用和定时调用
 - 16.3.3 映射应用程序主窗体
 - 16.3.4 显示消息框
- 16.4 测试脚本引擎
 - 16.4.1 文档对象
 - 16.4.2 创建全局对象容器
 - 16.4.3 初始化脚本引擎
 - 16.4.4 编辑脚本
 - 16.4.5 运行脚本
 - 16.4.6 演示用的脚本代码
- 16.5 部署脚本引擎
- 小结
- 第17章 开发Windows Service程序
 - 17.1 Windows Service概念介绍
 - 17.2 C#编写Windows服务的基本过程
 - 17.3 软件功能需求
 - 17.4 软件设计
 - 17.4.1 命令行参数设计
 - 17.4.2 数据库设计
 - 17.4.3 文件系统监视功能设计
 - 17.4.4 客户端设计
 - 17.5 软件说明
 - 17.5.1 客户端主界面 frmClient
 - 17.5.2 系统配置对话框 dlgConfig
 - 17.5.3 系统配置信息对象 MyConfig
 - 17.5.4 文件系统监视服务
 - 17.5.5 管理数据库连接
 - 17.5.6 启动程序
- 小结
- 第18章 文档对象模型
 - 18.1 文档对象模型定义
 - 18.1.1 文档
 - 18.1.2 结构化文档
 - 18.1.3 编程接口
 - 18.1.4 页面呈现
 - 18.2 理解文档对象模型
 - 18.2.1 XMLDom研究
 - 18.2.2 DOM概念内涵
 - 18.2.3 DOM概念外延
- 小结

章节摘录

版权页：插图：开发人员需要仔细设计相关算法，使得应用程序能够计算出尽可能小的脏矩形，使得这个脏矩形“多一分则太胖，少一分则太瘦”，刚刚好能够用于完整的维护用户界面而又避免浪费。正是这种斤斤计较提高了软件绘制用户界面的速度，缓解和避免了闪烁现象。

3.双缓存技术双缓存技术是解决用户界面闪烁的最方便可靠的方法。

闪烁是由于应用程序绘制用户界面过程缓慢造成的，但并不是应用程序所有的软件模块运行缓慢都会造成闪烁，仅仅是绘图过程缓慢才会造成闪烁。

开发人员可以在内存中建立一个位图图像用于缓存用户界面样式，然后所有的绘图操作都在这个内存中的位图中进行，绘图操作完成后就可以将这个位图图像快速输出到实际的显存中，从而显示到计算机显示器中。

应用程序在内存中的位图图像中绘制图形，相当于操作一个二维数组。

这只是应用程序内部的数据处理，不会导致显存数据的改变，也不会导致计算机屏幕像素值的改变，闪烁也无从谈起。

当绘图操作完成，内存中的位图图像准备完毕，应用程序就可以调用操作系统将位图数据填充到显存中。

这是一个很简单很高速的内存块复制过程，足以在一个显卡刷新周期内完成操作，应用程序“运筹于帷幄之中，决胜于千里之外”，从而避免了闪烁，双缓冲技术能够方便有效地解决闪烁问题，但这个技术不能滥用。

首先，双缓存技术实际上加大了应用程序绘制用户界面的工作量，整个绘图时间延长了，虽然没有闪烁，但用户界面响应比较迟钝。

其次，双缓存技术需要在内存中创建一个比较大的位图图像对象，这将额外占用大量的内存，是一种以空间换速度的做法，在某些情况下并不一定是最好的做法。

滥用双缓存技术最大的坏处就是它使人懒惰，它使得开发人员放松了对软件速度的要求，掩盖了软件性能问题。

这是一种“掩耳盗铃”的做法，不利于锻练出优秀的图形软件开发人员。

通常，在软件的开发过程中不可以使用双缓存技术，开发人员靠真本事解决用户一界面闪烁问题，只有等到软件对用户发布了才可以使用双缓存技术实现良好的用户体验。

这里讲的“开发人员的真本事”就是指设计合理的算法，精心的调试，以提高软件绘图速度来尽量降低闪烁。

编辑推荐

《魅力·实践·发现:精通C#编程》：展现精英高手发现之旅，站在最新平台开发实践，介绍流行软件神奇魅力。

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>