



全面系统，深入实用
通俗易懂，实例丰富

WPEF

专业编程指南

◆ 李应保 著



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

WPF

专业编程指南

李应保 著

电子工业出版社

Publishing House of Electronics Industry

北京·BEIJING

内 容 简 介

《WPF 专业编程指南》是一本 WPF 编程的专业参考书，全书通过大量的实例深入阐述了 WPF 中的传递事件、传递命令、相关属性、附加属性、逻辑树和视觉树等基本概念；介绍了各种画笔、画刷的使用方法；深入讨论了 WPF 中的各种控件以及这些控件在窗口或页面上的排版，并进而阐述了控件的风格和模板及数据绑定等相关技术。

本书对 WPF 中的图形系统及图形和排版的变换原理也进行了深入的探讨，并在此基础上讨论了 WPF 中的动画技术。多媒体不是 WPF 专有的技术，但本书介绍了在 WPF 中使用多媒体的实用技术。用户控件和自定义控件是 WPF 中比较深入的内容，本书最后两章对这一课题进行了深层次的研究，通过对 Ribbon 控件的开发，不仅可以了解开发用户控件和自定义控件的方法，而且可以体会 WPF 项目的组织及多种 WPF 技术细节的综合运用。

本书可供 .NET 桌面及互联网应用程序的开发人员、项目管理人员或准备进入这一领域的相关工程技术人员，以及大专院校相关专业的师生参考学习。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目 (CIP) 数据

WPF 专业编程指南 / 李应保著. —北京: 电子工业出版社, 2010.1
ISBN 978-7-121-10011-6

I. W… II. 李… III. 窗口软件, Windows Vista—用户界面—程序设计 IV. TP316.7

中国版本图书馆 CIP 数据核字 (2009) 第 220308 号

策划编辑: 袁金敏

责任编辑: 高洪霞

特约编辑: 顾慧芳

印 刷: 北京天宇星印刷厂

装 订: 涿州市桃园装订有限公司

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

开 本: 860×1092 1/16 印张: 34 字数: 828 千字

印 次: 2010 年 1 月第 1 次印刷

印 数: 4000 册 定价: 68.00 元 (含光盘 1 张)

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：(010) 88254888。

质量投诉请发邮件至 zlt@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：(010) 88258888。

前言

2008年在世界经济历史上是一个不平凡的一年，在这一年中，美国的五大投资银行有两家破产。金融危机席卷全球，美国的失业率在过去的一年从4%飙升到了10%，有些城市的失业率甚至高达20%；加拿大的就业数字也是惨不忍睹，多伦多的失业率剧升到了12%（2009年8月数据）。在这样暗淡的经济背景下，有关WPF的工作却在快速增长，过去两年，和WPF相关的工作职位在北美和欧洲每半年翻一番，WPF初级职位的薪水在5万美元到7万美元之间，高级职位在10万美元以上。一般来说，新的开发平台在中国的应用比欧美要晚2~3年，但随着互联网的普及和软件开发外包到中国，这一迟后时间正在缩短。可以预见，在未来几年内，和WPF相关的工作职位也会在中国快速增长；所以，学习WPF编程技术正当其时，拥有WPF技术必将使你在职场上更加挥洒自如，也就是说，**学习WPF就是获取新的工作机遇。**

WPF是基于.NET的新一代界面开发平台，它实现了桌面应用程序和互联网应用程序的统一编程，实现了程序员长期梦想的数据驱动用户界面，融合了动画、多媒体的功能，跨越了图形和控件、控件和排版等技术上的界限，在很短的时间内实现并超越了Flash和PDF的相关功能。近年来微软在软件开发上的投资额已和中等国家的GDP相当，其中.NET是微软最大的投资项目。在.NET 2.0之后，微软停止了WinForm的开发，而WPF实际上是微软在今后一段时间内唯一要不断投入的用户界面开发平台，这一平台在Vista、Windows 7 和Window Server上得到了广泛的支持。建立在WPF上的应用程序将会自动随着.NET的不断开发而自动拓展新的功能：把WPF和WCF（Windows Communication Foundation）技术相结合，可以很容易地实现面向服务的软件架构（SOA, Service Oriented Architecture）；WPF对图形流的支持，使得开发GIS应用程序更加方便；基于SilverLight和Ajax技术是互联网开发的新热点。所以，企业把应用程序的界面建立在WPF之上，不仅可以极大地缩短开发周期，而且可以把同一技术用在不同的项目上，从而极大地降低开发成本。比如过去开发桌面应用程序和互联网应用程序一直是两个不同的开发团队，而使用WPF，我们只需要一个开发团队。在过去的20年内，微软一直是用户界面开发的领跑者；若企业把应用程序建立在WPF之上，就不会担心落伍。所以，**应用WPF就是降低企业的开发成本。**

本书特点

本书深入浅出地介绍了WPF中的各种新概念，使用了大量图表和实例力图以整体的形式把WPF展现在读者的面前。笔者推崇Scott Meyers的写作风格（Scott Meyers的C++系列丛书在软件界有广泛的影响——笔者注），即以散文的笔调描述技术细节，以避免枯燥的叙述；希望读者在阅读本书时有一种像朋友在一起聚会的感觉，边喝啤酒边聊天，在聚会结束时，您会发现自己已经掌握了WPF技术。因此，**阅读本书是通向WPF专业编程的捷径。**

由于WPF是在.NET 3.0 之后引入的，故读者在使用WPF之前应该已经熟悉 .NET的编程环境、C#语言等基本知识。本书使用简单的UML（Unified Modeling Language）描述WPF类和类间的关系，若您具有UML的基本知识，对阅读本书会有帮助。书中的例子在Visual Studio 2008 和.NET FrameWork

3.5上调试过，有时笔者也使用了微软的Expression Blend 2.0调试，但后者不是必需的。

本书光盘使用说明

本书的配书光盘含有约100MB的源程序，所用的语言为C#和XAML。所有的例程在Visual Studio 2008 和.NET Framework 3.5上调试通过，笔者在创建某些例程时，使用过Microsoft Blend 2.0。Microsoft Blend工具在创建WPF界面时非常有用，但对于运行本书的例程不是必需的。本光盘的内容是对本书的补充，因书中着重介绍WPF编程模型和基本概念，光盘中则含有完整的源代码。

1. 光盘中的目录使用Yingbao.Chapterxx格式，如第1章的例程在Yingbao.Chapter1的目录下，第2章的例程在Yingbao.Chapter2的目录下等。全书共18章，整个光盘含有18个目录。

2. 在每个目录下有一个相应的Visual Studio解决方案文件，其文件名采用Yingbao.Chapterxx.sln格式，如第1章Visual Studio解决方案文件名为Yingbao.Chapter1.sln，第2章Visual Studio解决方案文件名为Yingbao.Chapter2.sln等。该文件中含有一个或多个项目，如Yingbao.Chapter13解决方案中含有六个项目。运行某个项目，您需要在Visual Studio中的Solution explore窗口下单击鼠标右键，在弹出菜单中选择“Set as StartUp Project”条目，然后，您只要按下“F5”功能键，或在Visual Studio的“Debug”菜单下选择“Start Debug”即可。

3. 例程中命名空间的名字采用Yingbao.Chapterxx的格式，如第1章所有例程中的类都在Yingbao.Chapter1命名空间中，第2章的所有例程中的类都在Yingbao.Chapter2命名空间中。虽然对于本书的例程来说使用不同的命名空间不是必需的，但对于组织大型项目，使用命名空间是良好的习惯。

4. 例程中的类名、属性名、域名、方法名采用通用电气（GE）编程规范，读者也可用其他的公司或自己公司的编程规范。

感谢

写作是一个费时费力的工作，笔者在写作本书的时候，得到了家人的支持和理解。电子工业出版社的杨福平副总编和袁金敏编辑对本书的出版做了大量的工作，笔者在此表示衷心感谢。

联系方式

最后，若您有什么建议和意见或者发现书中的错误，请和笔者联系：Yingbao.Li@gmail.com。

李应保

2009年9月8日于加拿大

目 录

第一篇 WPF编程基础

第 1 章 WPF 应用程序	2
1.1 WPF 应用程序的创建	2
1.2 Dos 窗口	3
1.3 WPF 应用程序的启动和终止	4
1.4 输入参数	5
1.5 在 Xaml 中创建 Application	7
1.6 窗口大小	10
1.7 互联网应用程序	10
1.8 应用程序的异常处理	11
1.9 应用程序中的资源	12
1.10 应用程序的发布	13
1.11 WPF 开发环境	14
1.12 本章小结	15
第 2 章 XAML 语言	16
2.1 XAML 是一种界面描述语言	16
2.2 XAML 的根元素	17
2.3 XAML 命名空间 (NameSpace)	17
2.4 XAML 和代码分离技术 (code behind)	18
2.5 子元素	19
2.6 相关属性 (Dependency Property)	20
2.7 附加属性 (Attached Property)	21
2.8 XAML 标记扩展	21
2.8.1 静态资源扩展 (StaticResourceExtension)	22
2.8.2 动态资源扩展 (DynamicResourceExtension)	23
2.8.3 数据绑定扩展 (Binding)	24
2.8.4 相对数据源扩展 (RelativeSource)	24
2.8.5 模板绑定 (TemplateBinding)	25
2.8.6 x:Type 扩展	26
2.8.7 x:Static 扩展	26
2.8.8 x:null 扩展	26
2.8.9 x:Array 扩展	26
2.9 本章小结	27

第 3 章 WPF 排版	28
3.1 排版基础	28
3.2 堆积面板 (StackPanel)	29
3.3 WrapPanel	34
3.4 停靠面板 (DockPanel)	35
3.5 表格式面板 (Grid)	39
3.5.1 设定 UI 元素在 Grid 中的位置	40
3.5.2 设定 Grid 行或列的尺寸	40
3.5.3 元素横跨多个行列时的设定	41
3.5.4 在 Grid 中保持多行或多列大小的一致性	44
3.6 UniformGrid	46
3.7 画布面板 (Canvas)	47
3.8 本章小结	48
第 4 章 WPF 中的属性系统	49
4.1 CLR 属性	49
4.2 相关属性的概念	50
4.2.1 相关属性的传递	50
4.2.2 WPF 对相关属性的支持	51
4.3 自定义相关属性	52
4.4 附加属性	58
4.5 本章小结	67
第 5 章 画笔和画刷	68
5.1 WPF 中的颜色	68
5.2 画刷	75
5.2.1 实心画刷 (SolidColorBrush)	76
5.2.2 梯度画刷 (GradientBrush)	77
5.2.3 线性梯度画刷 (LinearGradientBrush)	77
5.2.4 圆形梯度画刷 (RadialGradientBrush)	80
5.2.5 自制画刷 (DrawingBrush)	81
5.2.6 粘贴模式 (TileMode)	82
5.2.7 伸展方式 (Stretch)	83
5.2.8 图像画刷 (ImageBrush)	83
5.2.9 控件画刷 (VisualBrush)	85
5.3 画笔	88
5.4 本章小结	95

第二篇 WPF专业程序员必备

第6章 WPF 控制	98
6.1 WPF 控件概述	98
6.2 内容控件 (Content Control)	100
6.2.1 框架控件 (Frame)	100
6.2.2 WPF 按钮 (Button)	101
6.2.3 拨动按钮 (ToggleButton)	104
6.2.4 CheckBox 控件	104
6.2.5 RadioButton 控件	104
6.2.6 重复按钮 (RepeatButton)	105
6.2.7 带有标题栏的内容控件 (HeaderedContentControl)	106
6.2.8 分组框 (GroupBox)	107
6.2.9 伸展控件 (Expander)	109
6.2.10 标签控件 (Label)	110
6.2.11 为按钮设置热键	111
6.2.12 ToolTip	113
6.2.13 ScrollViewer	115
6.3 条目控件 (Items Controls)	116
6.3.1 菜单 (Menu)	117
6.3.2 工具条 (ToolBar)	123
6.3.3 Selector	126
6.3.4 组合框 (ComboBox)	126
6.3.5 TabControl	129
6.3.6 列表框 (ListBox)	132
6.3.7 ListView	135
6.3.8 状态条 (StatusBar)	138
6.3.9 树形控件 TreeView 和 TreeViewItem	140
6.4 文本控件 (Text Controls)	143
6.4.1 口令输入框 (PasswordBox)	143
6.4.2 文字输入框 (TextBox)	144
6.4.3 RichTextBox	145
6.5 范围控件 (Range Controls)	146
6.5.1 滚动条 (ScrollBar)	146
6.5.2 滑动条 (Slider)	147
6.5.3 进展条 (ProgressBar)	152
6.6 本章小结	152
第7章 传递事件和传递命令系统	153
7.1 WPF 中的元素树	153

7.2	传递事件 (Routed Event)	165
7.2.1	RoutedEventArgs	166
7.2.2	终止事件传播	166
7.2.3	处理传递事件	167
7.2.4	附加传递事件 (Attached Routed Event)	168
7.3	考察传递事件	168
7.3.1	键盘事件的产生和传递	174
7.4	自定义传递事件	174
7.5	管理键盘和鼠标输入事件	182
7.5.1	键盘输入	182
7.5.2	鼠标输入	182
7.6	传递命令	184
7.6.1	ICommand 接口	186
7.6.2	ICommandSource 接口	186
7.6.3	CommandTarget	186
7.6.4	命令绑定 (CommandBinding)	186
7.6.5	传递命令 (Routed Command)	187
7.6.6	WPF 命令仓库 (Command Repository)	187
7.7	本章小结	190
第 8 章	资源	191
8.1	资源定义及其类型	191
8.2	统一资源标识 (Unified Resource Identifier)	192
8.3	.NET 开发平台对资源国际化的支持	196
8.3.1	WinForm 下的资源管理	197
8.3.2	用 XAML 创建本地资源	200
8.4	WPF 元素中定义的资源	202
8.4.1	静态资源 (StaticResource)	203
8.4.2	资源的作用范围	204
8.4.3	静态扩展标识 (Static markup extension)	205
8.4.4	动态资源扩展标识 (DynamicResource Markup Extension)	208
8.5	本章小结	210
第 9 章	风格	211
9.1	Style 类	211
9.2	Setters	211
9.3	TargetType	215
9.4	BasedOn	218
9.5	触发器 (Triggers)	220
9.5.1	使用单一条件的触发器	221

9.5.2	使用多个条件的触发器	222
9.5.3	使用数据触发器 (DataTrigger)	223
9.6	风格中的资源	225
9.7	IsSealed	227
9.8	把风格定格定义在单独的文件中	227
9.9	在 FrameworkContentElement 中使用风格	228
9.10	再谈 Setter 属性	229
9.11	本章小结	230
第 10 章	模板	231
10.1	模板概述	231
10.2	控件模板	232
10.2.1	在控件中使用模板	232
10.2.2	在资源中使用模板	234
10.2.3	在控件模板中使用 TargetType	235
10.2.4	在模板中显示控件的内容	236
10.2.5	在模板中使用 ContentPresenter	237
10.2.6	模板中元素名 Name 属性	239
10.2.7	在模板中绑定控件的其他属性	239
10.2.8	使用模板显示电力系统的断路器和刀闸开关	240
10.2.9	在风格中使用模板	242
10.2.10	获取 WPF 控件的模板	243
10.3	数据模板 (DataTemplate)	244
10.3.1	我们所面临的问题	244
10.3.2	定义数据模板	247
10.3.3	在资源中使用数据模板	248
10.3.4	数据模板触发器	249
10.3.5	根据数据属性选择相应的模板	250
10.3.6	在数据模板中使用类型转换技术	253
10.4	ItemsPanelTemplate	258
10.5	层次结构数据模板 (HierarchicalDataTemplate)	259
10.6	本章小结	262
第 11 章	数据绑定 (Data Binding)	263
11.1	数据绑定概述	263
11.2	最简单的数据绑定——从界面元素到界面元素	264
11.2.1	一对一数据绑定	264
11.2.2	在 C# 中, 实现数据绑定	265
11.2.3	对不是 FrameworkElement 和 FrameworkContentElement 元素实现数据绑定	266
11.3	使用不同的绑定模式	266

11.4	动态绑定	267
11.5	最简单的数据绑定——从.NET 对象到界面元素	268
11.6	DataContext	271
11.7	控制绑定时刻	272
11.8	开发自己的 IValueConverter	273
11.9	在数据绑定中加入校验	275
11.9.1	开发业务规则类	276
11.9.2	在绑定中添加任意多个业务规则	279
11.9.3	在控件上显示校验信息	279
11.9.4	触发错误处理事件	280
11.9.5	清除控件上的错误信息	282
11.10	对集合对象的绑定	283
11.10.1	使用 DisplayMemberPath 属性	286
11.10.2	显示当前条目	286
11.10.3	遍历集合中的记录	288
11.10.4	增加或删除记录	290
11.10.5	对集合对象分组	293
11.10.6	对集合对象排序	294
11.10.7	对集合对象过滤	295
11.11	数据源	296
11.11.1	XML 数据源	296
11.11.2	.NET 对象数据源	301
11.12	层次结构数据的绑定	303
11.13	本章小结	303
第 12 章	窗口对话框和打印	304
12.1	窗口 (Window)	304
12.1.1	窗口的状态变化和事件	304
12.1.2	确定视窗的位置	309
12.1.3	确定视窗的大小	310
12.1.4	视窗状态属性 (WindowState)	310
12.1.5	视窗大小模式 (SizeMode)	310
12.1.6	视窗风格 (WindowStyle)	311
12.2	网页 (Page)	311
12.2.1	创建网页	312
12.2.2	KeepAlive 属性	312
12.2.3	NavigationService 属性	312
12.2.4	ShowsNavigationUI 属性	313
12.3	浏览窗口 (NavigationWindow)	313

12.3.1	使用统一风格	314
12.3.2	设置 NavigationWindow 的标题	314
12.3.3	浏览网页	315
12.3.4	使用 HyperLink 类	315
12.3.5	使用 NavigationService 转到不同的网页	318
12.3.6	使用浏览日志转换到不同的网页	319
12.3.7	浏览窗口的浏览事件	319
12.4	对话框 (DialogBox)	320
12.4.1	消息框 (MessageBox)	320
12.4.2	通用对话框	320
12.4.3	自定义对话框	322
12.5	打印输出	323
12.5.1	XPS 文档简介	323
12.5.2	创建 XPS 文档	324
12.5.3	显示 XPS 文档	328
12.5.4	打印	333
12.6	本章小结	333

第三篇 图形和动画

第 13 章	二维图形	336
13.1	WPF 图形系统概述	336
13.1.1	统一编程模型	336
13.1.2	坐标系统	338
13.1.3	Shape 和 Geometry	338
13.2	Shape 及其派生类	339
13.2.1	直线 (Line)	340
13.2.2	矩形 (Rectangle)	340
13.2.3	椭圆 (Ellipse)	341
13.2.4	折线 (Polyline)	341
13.2.5	多边形 (Polygon)	342
13.2.6	填充规则 (FillRule)	342
13.2.7	路径 (Path)	343
13.3	Geometry 及其派生类	343
13.3.1	直线 (LineGeometry)	344
13.3.2	矩形 (RectangleGeometry)	344
13.3.3	椭圆 (EllipseGeometry)	344
13.3.4	几何图形组 (GeometryGroup)	345

13.3.5	合并图形 (CombinedGeometry)	346
13.3.6	几何路径 (PathGeometry)	348
13.3.7	分段路径 (PathSegment)	350
13.3.8	弧线 (ArcSegment)	350
13.3.9	直线段 (LineSegment)	352
13.3.10	折线段 (PolyLineSegment)	353
13.3.11	柏之线 (BezierSegment)	353
13.3.12	多段柏之线 (PolyBezierSegment)	354
13.3.13	二次柏之线 (QuadraticBezierSegment)	354
13.3.14	多段二次柏之线 (PolyQuadraticBezierSegment)	355
13.3.15	迷你绘图语言	356
13.3.16	流几何图形 (StreamGeometry)	360
13.4	绘制 (Drawing)	361
13.4.1	使用 DrawingImage 显示几何图形	362
13.4.2	使用 DrawingVisual 来显示几何绘制	363
13.4.3	创建 DrawingVisual 宿主	363
13.4.4	绘制几何图形	364
13.4.5	把 DrawingVisual 对象加到 FrameworkElement 中的视觉树和逻辑树中	364
13.4.6	选择视觉元素 (Visual Hit Testing)	366
13.4.7	简单选择判断	366
13.4.8	多个视觉元素的选择判断	367
13.4.9	视觉元素重叠时的选择判断	367
13.5	本章小结	368
第 14 章	图形转换	369
14.1	图形转换概述	369
14.2	项目管理器	370
14.3	旋转转换 (RotateTransform)	376
14.4	位移转换 (TranslateTransform)	378
14.5	缩放转换 (ScaleTransform)	380
14.6	扭曲转换 (SkewTransform)	382
14.7	组合转换 (TransformGroup)	384
14.8	矩阵转换 (MatrixTransform)	385
14.8.1	矢量操作	385
14.8.2	H 坐标系	386
14.8.3	位移转换矩阵	387
14.8.4	旋转转换矩阵	388
14.8.5	缩放转换矩阵	388
14.8.6	扭曲转换矩阵	389

14.8.7	矩阵操作	389
14.9	本章小结	394
第 15 章	动画	395
15.1	WPF 中的动画	395
15.2	动画类继承树	396
15.3	一个简单的动画	397
15.4	控制动画	398
15.4.1	动画所用的时间 (duration)	399
15.4.2	设定动画开始时间 BeginTime	399
15.4.3	设定自动返回 (AutoReverse)	399
15.4.4	设定动画速度 (SpeedRatio)	400
15.4.5	加快和减慢动画 (AccelerationRatio 和 DecelerationRatio)	400
15.4.6	设定动画的重复特性 (RepeatBehavior)	402
15.4.7	设定动画的终止状态 (FillBehavior)	402
15.4.8	设定相关属性的动画范围 (From 和 To)	402
15.4.9	设定相关属性的动画范围 (By)	403
15.4.10	设定 IsAdditive 和 IsCumulative 属性	403
15.4.11	WPF 动画的时间片类	403
15.5	故事板 (Storyboard)	404
15.5.1	使用故事板的一般格式	404
15.5.2	设定 Target 和 TargetName	406
15.5.3	操作 Storyboard	406
15.6	KeyFrame	408
15.6.1	线性 KeyFrame	409
15.6.2	非线性 KeyFrame (Spline KeyFrame)	412
15.6.3	离散 KeyFrame (Discrete KeyFrame)	414
15.7	本章小结	416

第四篇 开发WPF产品

第 16 章	多媒体技术及其应用	418
16.1	播放.wav 声音格式的 SoudPlayer 和 SoundPlayerAction	418
16.1.1	装载.wav 文件	418
16.1.2	播放.wav 文件	419
16.1.3	停止播放	419
16.1.4	在 XAML 中使用 SoundPlayerAction	419
16.2	播放多种格式的声音和图像	420
16.2.1	播放模式	421

16.2.2	使用 MediaPlayer 实例	422
16.2.3	使用 MediaElement 和 MediaTimeline 实例	426
16.3	语音合成和语音识别	430
16.3.1	尝试 Windows Vista 的语音功能	431
16.3.2	使你的程序发音	432
16.3.3	PromptBuilder 和 SSML	433
16.3.4	语音识别中的语法	434
16.4	本章小结	436
第 17 章	定制控件和排版	437
17.1	用户控件和自定义控件	437
17.2	创建用户控件 (User Control)	439
17.2.1	设计用户控件 UI	439
17.2.2	开发支持用户控件 UI 的逻辑	442
17.3	创建自定义控件 (Custom Control)	446
17.4	创建自定义排版 (Custom Panel)	459
17.4.1	照片浏览器	461
17.5	本章小结	471
第 18 章	综合应用	472
18.1	Ribbon 界面概览	472
18.2	项目的组织	473
18.3	管理 Generic.XAML 文件	475
18.4	开发自定义控件	476
18.4.1	自定义控件间的关系	476
18.4.2	Ribbon 按钮	477
18.4.3	Ribbon 分组 (Group)	483
18.4.4	RibbonTabItem	492
18.4.5	RibbonApplicationMenuItem	493
18.4.6	RibbonApplicationMenu	495
18.4.7	RibbonQAToolBar	496
18.4.8	RibbonBar	497
18.4.9	RibbonWindow	498
18.4.10	支持不同皮肤	514
18.5	使用 Ribbon 自定义控件实例	516
18.6	本章小结	525
参考文献	526

第一篇 WPF编程基础

第 1 章 WPF 应用程序

第 2 章 XAML 语言

第 3 章 WPF 排版

第 4 章 WPF 中的属性系统

第 5 章 画笔和画刷

第1章 WPF应用程序

本章讨论WPF应用程序的创建，运行及退出时的技术细节。涉及创建WPF应用程序的三个重要的类：Application、Window和Page、应用程序的异常处理、发布等话题。

1.1 WPF应用程序的创建

在Windows操作系统中，所有的应用程序都在自己独立的进程中运行，WPF也不例外。每个进程都有自己独立的内存地址，进程间的数据是隔离的，进程之间不会相互影响。当应用程序创建进程时，同时创建一个或多个线程。线程在自己的进程空间中运行，在底层，WPF仍然使用Windows的消息驱动机制来实现事务处理。

创建WPF应用程序从创建Application类开始。WPF的主要功能是人机交互，我们可以把当代人机界面（UI）程序归为两大类：一类是桌面（desktop）应用程序；一类是互联网（Web）应用程序。WPF首次实现了对这两类应用程序的统一编程。

让我们来创建一个简单的WPF程序：

```
using System;
using System.Windows;
namespace Yingbao.Chapter1
{
    public class HelloWPF
    {
        [STAThread]
        public static void Main()
        {
            Window win = new Window();
            win.Title = "WPF application";
            win.Content = "Hello WPF!";
            win.Show();
            Application app = new Application();
            app.Run();
        }
    }
}
```

首先，用Yingbao.Chapter1声明命名空间，并将在本书的所有例程中使用这一惯例。第1章的例程在Yingbao.Chapter1的命名空间中，第2章的例程将在Yingbao.Chapter2的命名空间中等。

然后，引入了.NET中的两个命名空间：System和System.Windows。System命名空间提供.NET的基本服务，System.Windows中含有WPF的基本服务。Microsoft把WPF所用的类，都放在System.Windows的命名空间或其下面的命名空间中，如System.Windows.Controls、System.Windows.Input，等等。在Visual Studio的项目下，引入相应的Assembly，这样就可以使用这些命名空间中的类了。

Main函数是WPF程序的入口点，它必须是静态（static）的。所有WPF程序，都必须在单一线程公寓(STA)模型中运行。STA来源于COM，若对此概念不熟悉，可以参考COM的相关著作或文章。在