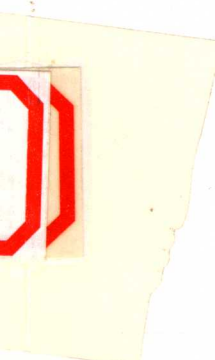


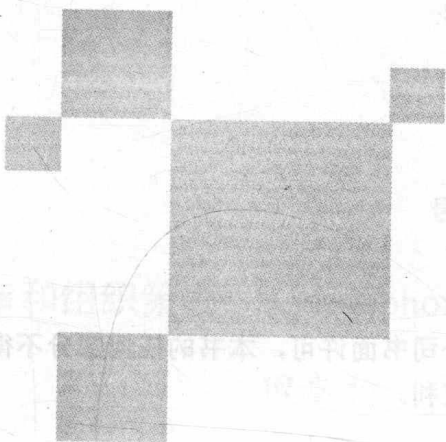
Microsoft®

Windows Presentation Foundation(WPF)应用开发

微软公司 著


 **人民邮电出版社**
POSTS & TELECOM PRESS





Windows Presentation Foundation(WPF)应用开发

微软公司 著

发刊特表

人民邮电出版社

北京

图书在版编目 (CIP) 数据

Windows Presentation Foundation(WPF)应用开发 /
微软公司著. — 北京 : 人民邮电出版社, 2011. 2
ISBN 978-7-115-23343-1

I. ①W… II. ①微… III. ①窗口软件,
Windows—程序设计 IV. ①TP316.7

中国版本图书馆CIP数据核字(2010)第172994号

版 权 声 明

本书的著作权归微软公司所有。未经微软公司书面许可, 本书的任何部分不得以任何形式或任何手段复制或传播。著作权人保留所有权利。

内 容 提 要

Windows Presentation Foundation (WPF) 为开发人员提供了统一的编程模型, 可用于构建合并了 UI、媒体和文档的丰富 Windows 智能客户端用户体验。本书介绍如何生成和配置 Windows Presentation Foundation (WPF) 的解决方案。

全书共 9 章, 讲述了如何生成 WPF 应用程序, 在 WPF 中创建 UI, 自定义 WPF 应用程序的外观, 绑定 UI 控件到数据源, 使用集合视图和数据模板显示数据, 实现用户控件和自定义控件, 创建、打包和打印文档, 使用 2D 和 3D 以及多媒体, 最后还讲解了如何配置和部署 WPF 应用程序。

本书适合具备 .NET Framework 应用程序开发技能, 想要使用 Windows Presentation Foundation 在应用程序中实现展示层应用的开发人员阅读。

Windows Presentation Foundation (WPF) 应用开发

- ◆ 著 微软公司
责任编辑 刘 浩
- ◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
邮编 100061 电子函件 315@ptpress.com.cn
网址 <http://www.ptpress.com.cn>
北京艺辉印刷有限公司印刷
- ◆ 开本: 787×1092 1/16
印张: 16
字数: 382 千字

2011 年 2 月第 1 版

2014 年 7 月北京第 5 次印刷

ISBN 978-7-115-23343-1

定价: 52.00 元 (附光盘)

读者服务热线: (010)67132692 印装质量热线: (010)67129223

反盗版热线: (010)67171154

编审和组织策划: Kyle Uphoff Foong Chee Ngiam 王 林

田本和 蒋 斌 梁 健

技术编审: 蒋 斌 张 充 孙 欣 宫 丽

前 言

Windows Presentation Foundation (WPF) 是下一代显示系统，用于生成能带给用户震撼视觉体验的 Windows 客户端应用程序。使用 WPF，您可以创建广泛的独立应用程序以及浏览器承载的应用程序。

通过学习本课程，读者能够编写使用 Windows Presentation Foundation 开发丰富的展示层应用。具体技能包括：

- 创建 WPF 应用程序；
- 在 WPF 应用程序中生成 UI；
- 自定义 WPF 应用程序的外观；
- 绑定 UI 控件到数据源；
- 绑定 UI 控件到集合；
- 在 WPF 应用程序中创建新控件；
- 在 WPF 应用程序中管理文档；
- 为 WPF 应用程序添加图形和多媒体支持；
- 配置和部署 WPF 应用程序。

本书结构

本书共分为 9 章。

第 1 章 “使用 Windows Presentation Foundation 创建应用程序”，讲述了如何生成 WPF 应用程序。

第 2 章 “生成用户界面”，讲述了如何在 WPF 中创建 UI。

第 3 章 “自定义外观”，讲述了如何使用样式、模板、触发器和动画自定义 WPF 应用程序的外观。

第 4 章 “数据绑定”，讲述了如何绑定 UI 控件到数据源。

第 5 章 “数据绑定到集合”，讲述了如何使用集合视图和数据模板呈现、排列以及分组数据。

第 6 章 “创建新控件”，讲述了如何实现用户控件和自定义控件。

第 7 章 “管理文档”，讲述了如何创建、打包和打印文档。

第 8 章 “图形和多媒体”，讲述了如何在 WPF 应用程序中使用 2D 图形、3D 图形以及多媒体。

第 9 章 “在 Windows Presentation Foundation 中配置和部署应用程序”，讲述了如何配置和部署 WPF 应用程序。

教学参考资料

获得微软授权使用该教材进行教学的教师，除了可以获得上述材料外，还可以从微软公

司获得如下教学资料。

教参	章节教参与教学大纲
实验与习题答案	实验的具体操作步骤和习题答案
虚拟机	配置好的虚拟机实验环境，用于进行实验和课堂演示
实验室安装指南	包括在实验室部署虚拟机的指南
PowerPoint	用于进行课堂演示

虚拟机

本书案例所提供虚拟机实验均采用微软的 Lab Launcher 进行操作,Microsoft Lab Launcher 是一种虚拟机操作界面，它让用户用一种更简易的操作界面进行虚拟机操作。该工具基于 Virtual Server R2 SP1 实现。虚拟机安装完毕后，用户即可使用此界面进行操作，操作界面参见图 0-1。

Lab Launcher 的运行环境要求如下。

- 操作系统为 Windows XP 简体中文版或 Windows Vista 简体中文版。
 - 系统中安装 .NET Framework 2.0 简体中文版。
 - 系统中安装 Microsoft Virtual Server 2005 R2 简体中文版。
 - 2GB 物理内存。
- 在本课程中，共包含一套虚拟机，分别对应不同的内容。

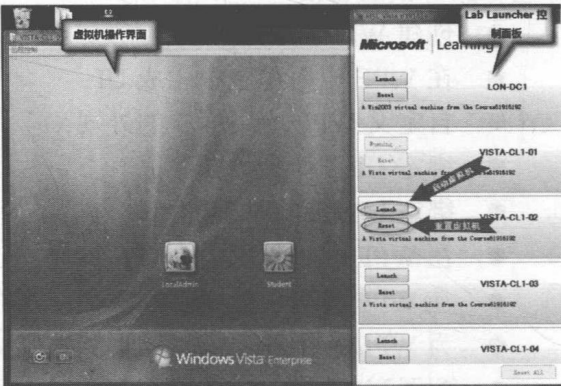


图 0-1 虚拟机操作界面

虚拟机	角色
10083A-LON-DEV-01	第 1 章的虚拟机
10083A-LON-DEV-02	第 2 章的虚拟机
10083A-LON-DEV-03	第 3 章的虚拟机
10083A-LON-DEV-04	第 4 章的虚拟机
10083A-LON-DEV-05	第 5 章的虚拟机
10083A-LON-DEV-06	第 6 章的虚拟机
10083A-LON-DEV-07	第 7 章的虚拟机
10083A-LON-DEV-08	第 8 章的虚拟机
10083A-LON-DEV-09	第 9 章的虚拟机

目 录

第 1 章 使用 Windows Presentation Foundation (WPF) 创建应用程序

1.1 WPF 概述	1
1.1.1 什么是 WPF	2
1.1.2 WPF 体系结构	3
1.1.3 在 WPF 中定义用户界面	3
1.1.4 WPF 的功能和特性	4
1.1.5 WPF 应用程序的类型	5
1.2 创建简单的 WPF 应用程序	5
1.2.1 演示：使用 Visual Studio 2008 创建 WPF 应用程序	5
1.2.2 定义应用程序	6
1.2.3 定义窗口或者页面	6
1.2.4 添加控件	6
1.2.5 生成和运行 WPF 应用程序	7
1.3 处理事件和命令	7
1.3.1 WPF 事件模型	7
1.3.2 处理 WPF 控件事件	7
1.3.3 什么是路由事件	8
1.3.4 定义路由事件	9
1.3.5 什么是命令	10
1.3.6 演示：定义命令	10
1.4 在页面间进行导航	11
1.4.1 WPF 导航模型	11
1.4.2 演示：使用超链接导航页面	12
1.4.3 处理页面导航事件	12
1.4.4 使用导航服务维持状态	13
1.5 实验：创建 WPF 应用程序	14
1.5.1 实验 1-1：创建一个独立 WPF 应用程序	14

1.5.2 实验 1-2：处理事件和命令	18
1.5.3 实验 1-3：在页面间进行导航	20
1.5.4 实验 1-4：创建一个 XBAP 应用程序	24
1.6 习题	27

第 2 章 生成用户界面

2.1 定义页面布局	33
2.1.1 WPF 页面布局模型	33
2.1.2 WPF 布局类	34
2.1.3 演示：使用面板定义布局	35
2.1.4 演示：使用网格定义布局	35
2.2 使用内容控件生成用户界面	35
2.2.1 什么是内容控件	36
2.2.2 演示：使用内容控件生成用户界面	37
2.2.3 什么是 Headered 内容控件	37
2.2.4 演示：使用 Headered 内容控件创建用户界面	38
2.3 使用项控件生成用户界面	39
2.3.1 什么是项控件	39
2.3.2 处理项选择	40
2.3.3 演示：使用项控件创建用户界面	41
2.4 承载 Windows 窗体控件	41
2.4.1 为什么要在 WPF 中承载 Windows 窗体控件	41
2.4.2 在 WPF 应用程序中引用 Windows 窗体控件	42

2.4.3	在 XAML 中使用 Windows 窗体控件	42	3.4.1	什么是触发器	68
2.4.4	与 Windows 窗体控件进行交互	43	3.4.2	定义属性触发器	69
2.5	实验：生成用户界面	44	3.4.3	什么是动画	69
2.5.1	实验 2-1：定义页面布局 和添加内容	44	3.4.4	定义动画	70
2.5.2	实验 2-2：使用项控件 增强用户界面	46	3.4.5	演示：使用触发器和动画增 强控件	72
2.5.3	实验 2-3：集成 Windows 窗体控件	49	3.5	实验：自定义 WPF 应用 程序的外观	72
2.6	习题	51	3.5.1	实验 3-1：在应用程序中 共享逻辑资源	72
第 3 章	自定义外观	54	3.5.2	实验 3-2：使用样式创建 一致的用户界面	74
3.1	在应用程序中共享逻辑资源	54	3.5.3	实验 3-3：使用控件模板 更改控件外观	75
3.1.1	什么是资源	54	3.5.4	实验 3-4：使用触发器和 动画增强用户界面	77
3.1.2	定义资源	54	3.6	习题	80
3.1.3	在 XAML 中引用资源	55	第 4 章	数据绑定	86
3.1.4	以编程方式引用资源	57	4.1	数据绑定概述	86
3.1.5	跨应用程序重用资源	58	4.1.1	WPF 数据绑定模型	86
3.1.6	演示：在 WPF 应用程序 中共享资源	59	4.1.2	绑定源和绑定目标	87
3.1.7	定义本地化资源	59	4.1.3	数据绑定模式	87
3.2	使用样式创建一致的用户界面	60	4.2	创建数据绑定	88
3.2.1	什么是样式	60	4.2.1	绑定到类属性	88
3.2.2	定义样式	61	4.2.2	绑定多个控件到 一个类中	90
3.2.3	扩展样式	62	4.2.3	绑定到整个对象	91
3.2.4	以编程方式设置样式	63	4.2.4	绑定到 XML 数据	91
3.3	使用控件模板更改控件外观	64	4.2.5	绑定到另一个用户界面 元素	93
3.3.1	什么是控件模板	64	4.2.6	演示：绑定到不同的 数据源	94
3.3.2	为内容控件定义控件 模板	64	4.3	实现属性更改通知	95
3.3.3	为项控件定义控件模板	65	4.3.1	什么是属性更改通知	95
3.3.4	使用模板绑定提供 用户定制	67	4.3.2	传递属性更改通知到 绑定目标	96
3.3.5	演示：使用控件模板更改 控件外观	68	4.3.3	传递值更改到绑定源	98
3.4	使用触发器和动画增强 用户界面	68	4.3.4	演示：触发源更新	99

4.4 转换数据.....99	5.3.1 什么是数据模板..... 136
4.4.1 默认数据转换.....99	5.3.2 定义和使用数据模板..... 136
4.4.2 实现一个自定义值 转换器.....100	5.3.3 定义数据模板为资源..... 137
4.5 验证数据.....102	5.3.4 使用数据模板中的 数据触发器..... 138
4.5.1 默认数据验证.....102	5.4 实验: 绑定用户界面 元素到集合..... 139
4.5.2 提供视觉验证反馈.....103	5.4.1 实验 5-1: 使用集合 视图呈现数据..... 139
4.5.3 定义一个自定义验证 规则.....104	5.4.2 实验 5-2: 使用数据 模板呈现数据..... 145
4.5.4 使用 XAML 指定验证 规则.....106	5.5 习题..... 146
4.6 实验: 实现数据绑定 应用程序.....106	第 6 章 创建新控件..... 153
4.6.1 实验 4-1: 创建数据 绑定.....106	6.1 控件创作概述..... 153
4.6.2 实验 4-2: 实现属性 更改通知.....110	6.1.1 为什么要创建新控件..... 153
4.6.3 实验 4-3: 转换数据.....115	6.1.2 创建新控件的选项..... 154
4.6.4 实验 4-4: 验证数据.....117	6.1.3 用户控件..... 154
4.7 习题.....120	6.1.4 自定义控件..... 155
第 5 章 数据绑定到集合.....124	6.1.5 FrameworkElement 派生控件..... 155
5.1 绑定到对象集合.....124	6.2 创建控件..... 156
5.1.1 绑定到集合概述.....124	6.2.1 创建用户控件..... 156
5.1.2 什么是 Observable Collection.....125	6.2.2 实现属性和事件..... 159
5.1.3 定义一个 Observable Collection 类.....126	6.2.3 创建自定义控件..... 162
5.1.4 LINQ 简介.....128	6.2.4 实现命令..... 163
5.1.5 绑定到 ADO.NET 数据对象.....130	6.2.5 使用主题增强控件..... 167
5.2 使用集合视图呈现数据.....131	6.2.6 演示: 实现一个 NumericUpDown 控件... 168
5.2.1 什么是集合视图.....132	6.3 实验: 使用自定义控件 增强用户界面..... 168
5.2.2 创建和使用集合视图.....132	6.3.1 实验 6-1: 实现一个 自定义控件..... 168
5.2.3 使用集合视图排列数据...133	6.4 习题..... 172
5.2.4 使用集合视图筛选数据...133	第 7 章 管理文档..... 176
5.2.5 使用集合视图分组数据...134	7.1 创建和查看流文档..... 176
5.2.6 创建主从用户界面.....135	7.1.1 什么是流文档..... 176
5.3 使用数据模板呈现数据.....135	7.1.2 定义一个流文档..... 177

7.1.3	流文档容器类型	177	支持	203	
7.1.4	演示: 定义 FlowDocument 容器	178	8.1.2	绘制形状	204
7.1.5	有关流的控件	178	8.1.3	什么是路径和几何图形	204
7.1.6	自定义文本	179	8.1.4	演示: 填充形状和几何图形	205
7.1.7	演示: 自定义 FlowDocument 中的文本	180	8.1.5	演示: 使用和动画化转换	205
7.2	创建和查看固定文档	180	8.2	显示图像	205
7.2.1	什么是固定文档	181	8.2.1	WPF 图像处理组件	205
7.2.2	定义固定文档	181	8.2.2	演示: 在 WPF 中显示图像	206
7.2.3	定义一个固定文档查看器	182	8.2.3	编码和解码图像	207
7.2.4	演示: 在 XAML 中创建一个 FixedDocument	182	8.2.4	旋转、转换和裁切图像	208
7.3	打包文档	183	8.3	创建 3D 图形	209
7.3.1	打包文档的支持	183	8.3.1	2D 与 3D 的区别	209
7.3.2	打包文档部件为 ZIP 文件	183	8.3.2	WPF 中的 3D 图形支持	210
7.3.3	数字签名内容	185	8.3.3	什么是 Viewport3D	210
7.3.4	包和部件与信息之间的关联	186	8.3.4	Viewport3D 的照相机类型	211
7.3.5	演示: 创建一个数字签名 ZipPackage	187	8.3.5	创建模型	212
7.4	打印文档	187	8.3.6	演示: 呈现 3D 内容	213
7.4.1	什么是 XML 纸张规范	187	8.4	操作 3D 环境	213
7.4.2	演示: 打印文档	187	8.4.1	为 3D 模型指定材料	213
7.4.3	控制打印任务	188	8.4.2	为 3D 模型指定光	214
7.4.4	管理打印队列	189	8.4.3	演示: 转换 3D 模型	214
7.5	实验: 管理文档	191	8.4.4	演示: 3D 模型动画制作	215
7.5.1	实验 7-1: 创建和显示流文档	191	8.5	添加多媒体	215
7.5.2	实验 7-2: 打印文档	194	8.5.1	WPF 对多媒体的支持	215
7.5.3	实验 7-3: 创建和签名 XPS 文档	195	8.5.2	媒体播放模式	215
7.6	习题	196	8.5.3	使用 MediaElement 显示媒体	216
第 8 章	图形和多媒体	203	8.5.4	控制 MediaElement 的操作	216
8.1	创建 2D 图形	203	8.5.5	使用 MediaPlayer 播放媒体	216
8.1.1	WPF 中的 2D 图形		8.6	实验: 图形和多媒体	217
			8.6.1	实验 8-1: 显示 2D	

图形	218	9.2.3 安装 .NET Framework	231
8.6.2 实验 8-2: 显示图像	219	9.3 部署 XBAP 应用程序	232
8.6.3 实验 8-3: 显示 3D 图形	220	9.3.1 必须部署的 XBAP 文件	232
8.6.4 实验 8-4: 播放视频 剪辑	221	9.3.2 什么是清单文件	232
8.7 习题	223	9.3.3 发布一个 XBAP 应用 程序到 Web 服务器	233
第 9 章 在 WPF 中配置和部署 应用程序	227	9.3.4 安装一个 XBAP 应用 程序到客户端计算机	233
9.1 部署选项	227	9.4 配置清单设置	234
9.1.1 部署技术	227	9.4.1 使用清单生成和 编辑工具	234
9.1.2 完全信任和部分信任 应用程序	228	9.4.2 Mage.exe 命令	234
9.1.3 部分信任中可用的 WPF 功能	228	9.4.3 为新建和更新文件 设置选项	235
9.1.4 部分信任中不可用的 WPF 功能	228	9.4.4 为签名文件设置选项	235
9.1.5 本地 Intranet 上的 XBAP 应用程序	229	9.5 实验: 配置和部署 WPF 应用程序	236
9.2 部署独立的 WPF 应用程序	230	9.5.1 实验 9-1: 部署一个独立的 WPF 应用程序	236
9.2.1 使用 Windows Installer 部署 一个独立的应用程序	230	9.5.2 实验 9-2: 更新一个 部署清单	239
9.2.2 使用 ClickOnce 部署一个 独立的应用程序	230	9.5.3 实验 9-3: 部署一个 XBAP 应用程序	240
		9.6 习题	242

第 1 章 使用 Windows Presentation Foundation (WPF) 创建应用程序

在过去的 10 年中，应用程序的开发模型产生了两个分支，那就是传统的“智能客户端”安装应用程序和基于 Web 的应用程序，两者都为了能在开发界中占据统治地位而展开竞争。这两个模型都要求开发人员牺牲在另一个模型中被视为理所当然的功能。

Web 应用程序提供了更大的作用范围、更简单的部署等功能，但其代价是状态丢失及开发模型更复杂等。同样 Web 应用程序也不能充分利用现有计算机硬件平台丰富的图形图像和多媒体效果。

另一方面，Windows 应用程序确保了应用程序可以脱机工作，并且可以充分利用客户端硬件。但是，也失去了易于部署的特点。同样，Web 开发领域所拥有的大量设计软件和可视化开发工具，也无法用于 Windows 应用程序的界面开发。但显而易见的是，客户端应用程序模型可以为独立或基于浏览器的应用程序提供更好的硬件支持，以实现更复杂的功能。

自从在 Windows 的第一个版本中首次出现以来，构成 Windows 用户体验核心的图形子系统（User32 和 GDI 库）实际上已经诞生了将近 20 年。当然，随着时间的推移，User32 和 GDI 库经历了重大的演变和不断的发展，并且在所有领域都引入了很多新的服务和功能。同时，还出现了诸如 Direct3D 这样的新呈现技术，这些技术能够充分利用在最新的视频卡中公开的图形功能。但是，目前的大多数主流应用程序都没有提供这些图形卡实现的体验。

Windows Presentation Foundation（以前称为“Avalon”）是 Windows 中新的关键图形子系统，它为用户界面、2D 和 3D 图形、文档和媒体提供了统一的方法。它在 .NET Framework 基础上生成，并利用 Direct3D 进行基于向量的呈现，为应用程序提供了强大的解决方案。另外，Windows Presentation Foundation（以下简称 WPF）引入 XAML（可扩展应用程序标记语言）。这是一种基于 XML 的语言，用于实例化和填充嵌套对象层次结构。这使设计人员和开发人员可使用诸如 Expression “Sparkle” 这样的工具以及第三方专家工具（包括 ZAM 3D 和 Mobiform Aurora），相互协作来完成客户端应用程序的设计和开发。在开发过程中，设计人员可以专注于界面设计而无需关心程序逻辑。

本章讲述了 WPF 的定义、作用和应用基础。

1.1 WPF 概述

WPF 是基于 Windows 操作系统用于创建客户端应用程序的新一代演示系统，可以用它创建从简单的文字处理器或媒体播放器，到企业级业务应用程序等一系列基于 Windows 的应用程序。图 1-1 所示为使用了 WPF 技术构建的范例应用程序界面。

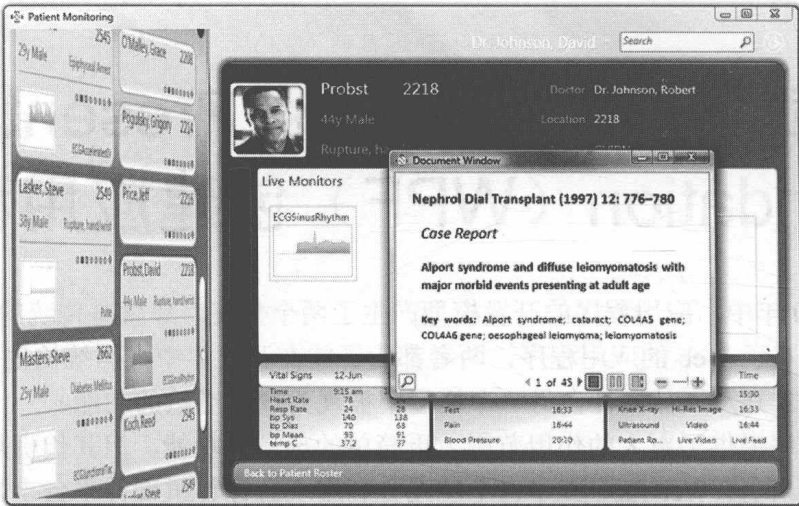


图 1-1 WPF 范例程序

1.1.1 什么是 WPF

WPF 是生成基于 Windows 的应用程序的新基础（Foundation），可以使用媒体、文档和应用程序用户界面。使用 WPF，既可以创建独立的应用程序，也可以创建基于浏览器运行的应用程序。

WPF 的核心是一个与分辨率无关并且基于向量的呈现引擎，能充分利用现代图像硬件的优势。

WPF 是基于现有的 Microsoft .NET Framework 2.0 版本，并与以下 3 项新技术一起构成了 Microsoft .NET Framework 3.0 版本（原来的 WinFX）的部分发布内容，如图 1-2 所示。

- Windows Communication Foundation (WCF): 该项技术可以让开发者生成并运行互联系统。
- Windows Workflow Foundation (WF): 该项技术可以让开发者更加方便地将 workflow 以及进程添加到应用程序中。
- Windows CardSpace (WCS)身份选择器: 该项技术可以让用户简单、安全、信任地使用他们的数字身份进行在线服务。

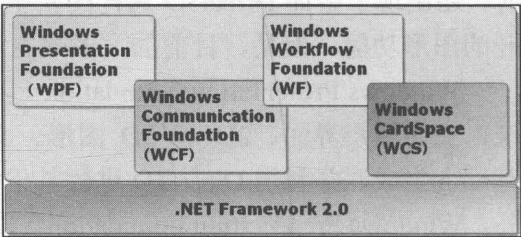


图 1-2 WPF 和其他 Foundation 的关系

通过使用 .NET Framework 3.0 版本中这些技术，可以创建各种不同类型的应用程序，比如使用 WPF 界面创建单用户系统和大型企业系统，使用 WCF 创建跨境通信系统，使用 WF 创建声明式业务流程，或者使用 WCS 创建在线信息管理程序。



问题与思考

你曾经使用过哪些其他的用户界面框架呢？

1.1.2 WPF 体系结构

WPF 是通过基于 .NET Framework 2.0 版本的托管代码公开其编程模型的。WPF 包含下列 3 个主要组件 (如图 1-3 所示):

- PresentationFramework;
- PresentationCore;
- milcore。

其中, PresentationFramework 和 PresentationCore 都是托管组件, 而 milcore 是非托管组件。Microsoft DirectX 应用程序接口 (API) 能够启用 WPF 所有的显示处理, 而且提供充足的软硬件渲染。由于 milcore 组件是以非托管代码编写的, 所以它可以提供更好的性能, 而且能够与 DirectX 引擎进行更为紧密的集成。

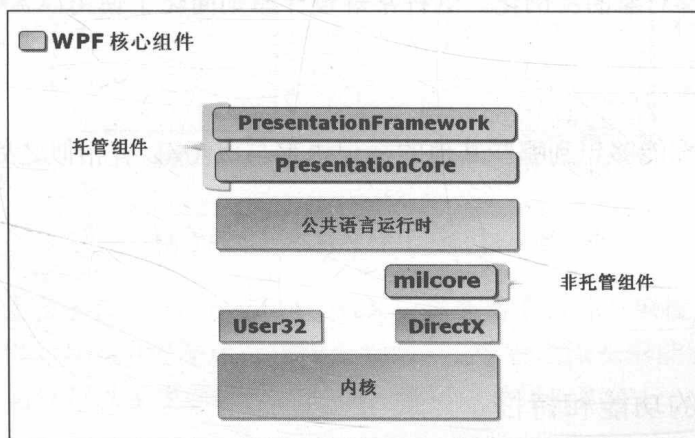


图 1-3 WPF 核心组件

PresentationFramework 和 PresentationCore 能够提供所有在应用程序中用得着的类和 API。

1.1.3 在 WPF 中定义用户界面

可以使用可扩展应用程序标记语言 (XAML) 中的声明代码生成用户界面, 也可以在 Microsoft Visual C# 或 Microsoft Visual Basic 中编写命令式代码来生成用户界面。

在大多数情况下, 采用这两种方法形成的结果都是相同的, 但是使用 XAML 更加易于编写和维护。对于用户界面来说, 没有什么不能在 XAML 和代码中完成的。

XAML 是一种用于声明性应用程序编程的标记语言。可以用 XAML 创建自己的用户界面定义, 并将应用程序的 XAML 保存在后缀名为 .xaml 的文件中。

图 1-4 显示了一段 .xaml 文件的代码及其执行效果。

利用 XAML, 通过使用 .NET Framework 编程模型可以使得创建用户界面 (UI) 更加方便, 因为只需要在声明性 XAML 标记中创建用户界面元素, 然后将用户界面定义与应用程序逻辑分离开来即可。XAML 直接描述了托管对象的实例。

所有能够在标记语言中完成的事情, 都可以在代码中完成。因此, 可以不使用 XAML,

而是直接使用代码来创建自己的用户界面。在 Windows 窗体中，用户界面的序列化格式就是代码，但是在 WPF 中却是 XAML。可以将 XAML 简单描述为 WPF 类的序列化格式。

XAML 简化了为 .NET Framework 编程模型创建 UI 的过程。在声明性 XAML 标记中创建可见的 UI 元素，然后使用代码隐藏文件（通过分部类定义与标记相连接）将 UI 定义与运行时逻辑相分离。在 XAML 中混合代码和标记的功能很重要，因为 XML 本身是声明性的，不会为流控制真正建议一个模型。基于 XML 的声明性语言非常直观，可以为用户（尤其是具有 Web 设计和技术背景的人员）创建从原型到生产的各种界面。与其他大多数标记语言不同，XAML 直接呈现托管对象的实例化。这种常规设计原则简化了使用 XAML 创建的对象代码和调试访问。

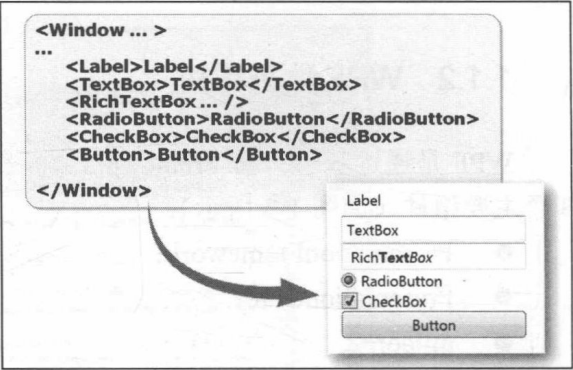


图 1-4 XAML 代码效果



你能够想到哪些其他的标记语言与 XAML 有相似之处呢？

1.1.4 WPF 的功能和特性

WPF 作为 .NET Framework 类型的一个子集存在，这些类型大多位于“System.Windows”命名空间。如果用户以前已使用 .NET Framework 通过诸如 ASP.NET 和 Windows 窗体之类的托管技术生成应用程序，那么应该熟悉 WPF 的基本编程体验；可以使用 .NET Framework 编程语言（如 C#或 Visual Basic）实例化类、设置属性、调用方法以及处理事件。WPF 提供以下功能和特性：

- 基于 XAML 的用户界面；
- 页面布局管理；
- 数据绑定；
- 2D 和 3D 图形；
- 多媒体；
- 动画；
- 文档和打印；
- 安全；
- 可访问性；
- 本地化；
- 与 Windows 窗体控件的互操作性。

1.1.5 WPF 应用程序的类型

当想要创建传统的基于 Windows 的应用程序，例如文字处理器或者业务应用程序来部署到个体客户端计算机的时候，就可以使用独立应用程序类型。而当想要创建能够在 Web 浏览器中运行的应用程序时，则可以使用 XAML 浏览器应用程序(XBAP)。独立应用程序和 XBAP 的界面如图 1-5 所示。

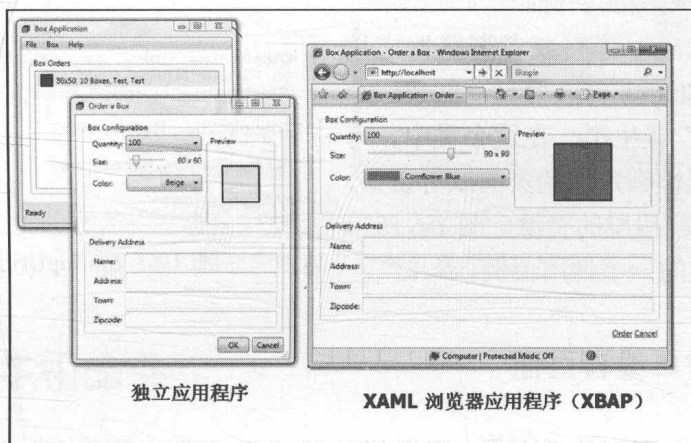


图 1-5 独立应用程序与 XAML 浏览器应用程序

XBAP 更形象地代表了传统 Microsoft ActiveX，或是基于组件对象模型 (COM) 并承载在浏览器模型中的控件。当应用程序运行权限受到限制，或者当浏览器能够承载该应用程序，或者当主要目的是简化分布以及更新模型的时候，都可以使用 XBAP。



问题与思考

你创建的应用程序中，编写为 XBAP 比编写为独立应用程序哪些更好？

1.2 创建简单的 WPF 应用程序

当想要创建一个 WPF 应用程序的时候，可以使用 Microsoft Visual Studio 2008 开发系统。Visual Studio 2008 提供了大量帮助信息例如项目、文件模版、代码实例、Microsoft IntelliSense 技术、WPF 设计器以及 XAML 编辑器以改进 WPF 开发体验。

本节讲述了当创建一个 WPF 应用程序时，Visual Studio 2008 会生成哪些内容。

1.2.1 演示：使用 Visual Studio 2008 创建 WPF 应用程序

在本演示中，将看到如何：

- 创建独立的 WPF 应用程序；
- 创建浏览器应用程序；
- 添加控件到应用程序。

1.2.2 定义应用程序

Visual Studio 会生成 XAML 应用程序文件，它会指定应用程序的代码后置类、起始窗口或者页面、应用程序范围的资源。

当创建一个独立应用程序或者浏览器应用程序的时候，Visual Studio 2008 会自动创建一个自定义“Application”派生类。在这个类中，可以使用“StartupUri”属性指定起始页面或者窗口，同时也可以指定任意应用层的资源。图 1-6 所示是“StartupUri”属性的一个使用范例。

```
<Application xmlns: x=... xmlns=...
x: Class="MyApp.App"
StartupUri="Window1.xaml">
  <Application.Resources>
    ...
  </Application.Resources>
</Application>
```

图 1-6 StartupUri 属性示例代码

1.2.3 定义窗口或者页面

独立应用程序包含窗口或者页面。它们都以 XAML 文件中的“<Window>”或“<Page>”元素表示出来。代码后置文件包含事件处理程序代码。

一个 Window 是最高级别的窗口，而一个 Page 是浏览器承载的页面。WPF 独立应用程序可以包含窗口和页面，但是 WPF 浏览器应用程序只包含页面。图 1-7 所示是“<Window>”和“<Page>”元素的范例代码。

```
<Window xmlns: x=...
xmlns=...
x:
Class="MyApp.Window1"
Title="My Window">
  <Grid>
    ...
  </Grid>
</Window>
```

```
<Page xmlns: x=...
xmlns=...
x: Class="MyApp.Page1"
WindowTitle="My Page">
  <Grid>
    ...
  </Grid>
</Page>
```

图 1-7 Window 和 Page 元素的范例代码

1.2.4 添加控件

WPF 附带了许多几乎可以在所有 Windows 应用程序中使用的常见 UI 组件，其中包括“Button”、“Label”、“TextBox”、“Menu”和“ListBox”。窗口和页面包含控件，控件会以 XAML 元素表示。这些元素的实例有“<Button>”和“<TextBox>”所有的控件元素，例如“<TextBox>”或“<Button>”都与类相似。可以在独立应用程序和浏览器应用程序中都使用相同的控件。图 1-8 所示是窗口和页面包含控件的范例代码。

```
...
<Grid>
  <TextBox Name="TextBox1" />
  <Button Name="Button1">Click here</Button>
</Grid>
...
```

图 1-8 控件代码范例