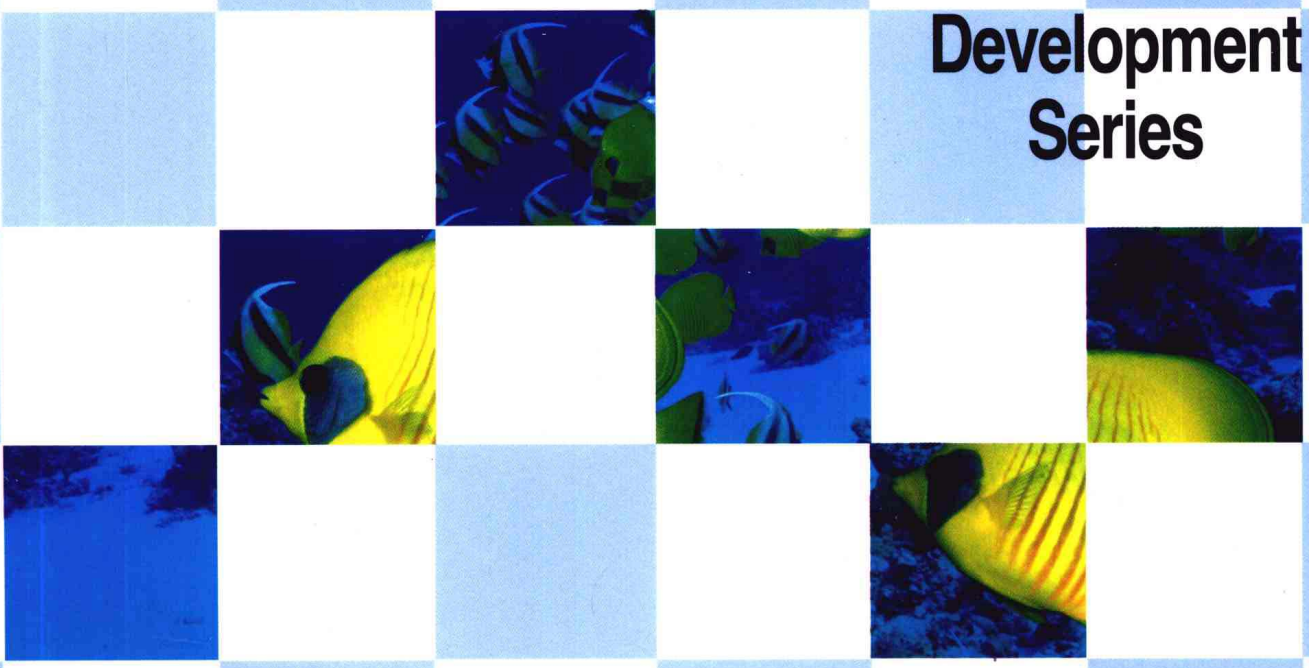
PEARSON
Addison Wesley

WPF核心技术

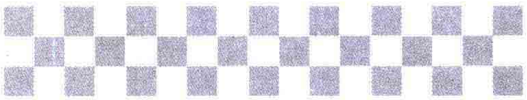
Essential Windows Presentation Foundation

[美] Chris Anderson 著
朱永光 译

Microsoft® .net™ Development Series



人民邮电出版社
POSTS & TELECOM PRESS



WPF核心技术

[美] Chris Anderson 著
朱永光 译

人民邮电出版社
北京

图书在版编目（CIP）数据

WPF核心技术 / （美）安迪生（Anderson, C.）著；朱永光译. —北京：人民邮电出版社，2009.6
ISBN 978-7-115-20662-6

Ⅰ. W… Ⅱ. ①安…②朱… Ⅲ. 窗口软件，Windows Vista—程序设计 Ⅳ. TP316.7

中国版本图书馆CIP数据核字（2009）第044147号

WPF 核心技术

◆ 著 [美] Chris Anderson
译 朱永光
责任编辑 刘映欣

◆ 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
北京顺义振华印刷厂印刷

◆ 开本：800×1000 1/16
印张：21.75
字数：525 千字 2009 年 6 月第 1 版
印数：1 – 3 000 册 2009 年 6 月北京第 1 次印刷

著作权合同登记号 图字：01-2007-5290 号
ISBN 978-7-115-20662-6/TP

定价：45.00 元

读者服务热线：(010)67132705 印装质量热线：(010)67129223
反盗版热线：(010)67171154

内容提要

本书围绕 WPF 中的一些重要概念进行详细的讲解，涉及 WPF 的各个方面，包括：WPF 的设计原则、应用程序的结构、内置控件、界面的布局、可视化效果、资源与数据、动作的操作、样式的处理和基础服务。本书还通过丰富的示例代码介绍了一些非常有用的 WPF 开发技巧。

本书适合各类使用 WPF 开发应用程序界面的技术人员阅读，不管是构架师、开发人员还是设计人员，都可以从中获益。

迈向 UI 的未来

1993 年第一次接触电脑编程，那时候只有控制台的界面，程序的输入输出都是文本形式的。后来使用 QB（QuickBASIC），可以通过很复杂的函数来绘制简单的界面。之后接触 FoxBASE，通过简单的函数调用就可实现下拉菜单，对此我还记忆犹新。

到了 1996 年，我开始使用 VB，要创建 Windows 程序的用户界面已经非常容易了。不过那个时候，提供了界面开发的微软技术还有 MFC 等。尤其是随着 ASP 等动态网页技术的出现，用户界面技术更是分离成了 Windows 和 Web。Web 的界面开发并不像 VB 那样轻松，当时也没有如今这么多的 JavaScript 库来辅助界面逻辑的开发。期间微软公司还推出了 Active Document 这样的技术，希望把 Windows 的界面技术带到 Web 中，甚至在 Visual Studio 6 中还加入了 ASP 开发的支持，不过并未真正解决问题。在这种情况下，我对朋友说，“终有一天微软公司会推出一种技术和工具，可以让 ASP 的界面开发像 VB 那样简单”。

于是，在 2001 年微软公司宣布了.NET 1.0，我们有了 ASP.NET Web Forms 技术。尽管如此，Windows 和 Web 的界面开发还是具有很大区别，开发人员很难把 Windows 的经验带到 Web 中。而且更糟糕的是，微软公司为大家带来 Web Forms 的同时，也带来了 Windows Forms，它和之前的 Windows 界面开发技术又有所不同，这样我们的界面技术更混乱了。另外，所有的界面技术还把开发人员和设计人员紧密地绑定到了一起。以此同时，Web 技术的发展日新月异，Web 的表现力越来越强，Web 界面对用户体验的影响越来越大。

如何解决不同界面技术并存的问题？如何既为用户提供强大炫目的界面，又能快捷地实现复杂的界面逻辑控制？微软公司在 2006 年推出的 WPF（Windows Presentation Foundation）给这些问题做出了很好的回答。

我相信 WPF 是微软界面开发技术的一个重要里程碑，对未来的界面技术有着深远的意义。我更相信，WPF 不是微软界面技术的终结，而是新时代的开始。从 WPF 开始，我们可以看到更符合人体工学的交互界面，Microsoft Surface 就是一个很好的例子。而从 WPF 开始，我们更会迎来自然界面（Nature UI）的时代——希望我们能一起迈向 UI 的未来！

上面就是我翻译本书的重要原因。虽然翻译书籍不是一件轻松的事情，尤其在翻译过程中还经历了汶川地震给我们带来的巨大震撼和诸多影响，但想到 WPF 本身的重要意义，我还是坚持到了最后。当然，本人翻译水平有限，书中难免有不妥之处，希望读者朋友能谅解，并提出意见。

朱永光

2009 年 4 月

对本书的赞誉

“作为 WPF 背后的构架师之一，Chris Anderson 巧妙地向我们既解释了‘如何’也解释了‘为何’。本书是那些希望了解 WPF 设计原则和最佳实践的人的优秀学习资料。”

——Anders Hejlsberg，微软公司技术伙伴

“如果 WPF 代表着 Windows 的下一代用户界面技术，那么 Chris Andersion 就代表着像 Charles Petzold 那样的 Windows 用户界面开发人员下一代领袖。”

——Ted Neward，TheServerSide.NET 创始编辑

“这是一本非常优秀的书籍，完成了向大家介绍 WPF 的重要工作，阐述了如何利用它所提供的那些强大功能。”

——Scott Guthrie，微软公司开发部门总经理

“WPF 是创建 UI 应用程序的全新技术，它汲取了来自 Windows Forms 和 Web 的设计原则。Chris 完成了一个重要的工作，不仅解释了如何使用这些 WPF 的新特性和功能（附带着大量代码和 XAML 标记），而且解释了它们为何如此工作。作为 WPF 的构架师，Chris 让我们见识到 WPF 的内部机理和设计原则，以及编写代码的技艺。假如你打算成为一名专业的 WPF 开发人员，本书提供的知识对你是极其重要的。”

——Brian Noyes，IDesign Inc.首席构架师，微软区域技术传播者，微软 MVP

“我有幸阅读了本书，发现它是非常有价值的学习资源，这是我诚心向别人推荐它的一个理由。我说一下我自己的一些做法，在面对新技术的时候，我总是希望了解它如何与被它取代的技术相关联并协同工作的。Chris 把 WPF 和 C++的 Windows 32 UI 进行了比较，以此作为他全书的开篇。Chris 不仅展示了他对驱使 WPF 运转的底层逻辑和工作原理的深刻理解，同时也展示了基于读者的知识体系来帮助读者的技巧，本书提供了一些例子可供读者学习如何开发高端应用程序。”

——Bill Seldon，核心开发工程师，InterKnowlogy

序 1

难以置信我是如何度过那段痛苦的时间的，最终，.NET 3.0 和 Windows Vista 还是成功地被发布了。

我还清楚地记得 Chris 是如何尽力为 PDC2003 的主题演讲者——Jim Allchin 艰苦地在后台准备.NET 3.0（那时被称为 WinFX）的第一个现场演示。那是一个压力格外巨大的主题演讲，由于洛杉矶当时正遭受森林火灾，Chris Anderson 的航班被迫取消。幸运的是，Chris Sells 已经到达了，假如 Chris Anderson 不能（实际上确实如此）准时到达洛杉矶的话，他就准备代替他做一些预备和演示。当时，Chris Anderson 在微软的工作就是保证 Windows Vista——包括 Windows Presentation Foundation（WPF）能出色且成功地被发布。但让他没有想到的是，大家花费 4 年时间才能让产品实际发布（这也是其获得成功的首要条件）。

那么，WPF 到底是什么东西呢？

正如其.NET 3.0 技术的姐妹——Windows Workflow Foundation（WF）那样，WPF 使用全面完整的方式来支持软件开发，它通过使用 XAML 以允许具有不同技能的人们能够在开发过程中协同工作。对于 WF 而言，XAML 让高级的过程和规则描述集成到由 C#或 Visual Basic 所编写的命令式代码中。而对于 WPF，XAML 在编写代码的人和那些设计又酷又帅的外观的人员之间搭建了桥梁。

WPF 真的是一项令人印象深刻的技术：文档、窗体和多媒体都能很好地被包装到一个以标记和代码构成的程序包中。

让我印象更深刻的事情是，Chris 把他每日工作的一些感受都融入到了本书当中，他以浅显易懂的方式把 4 年多的 WPF 方面的经验（包括截图）都介绍给了大家。

在这几年当中，对于 WPF 细节的问题，我非常有幸和 Chris 进行了多次交谈——有时在电话里，有时在他的办公室（和我的办公室相隔一个大厅）里，而有时是在扑克桌上。

这本书让我获益匪浅。

现在，它已经出版了，让我们来一饱眼福吧！

Don Box 华盛顿州，Yarrow Point

2007 年 1 月

序 2

在我进入微软之前，还没有见过太多像 Chris Anderson 这样牛的人。

我现在已经在微软工作了（实际上，我就和 Chris 隔着两道门），不过在此之前的很长一段时间里，我都在一家 Windows 开发人员培训公司当讲师。我和我的同事由一个善于思考的博士研究生所领导，他把在学术研究中的严谨态度也用于工作之上，对每个问题都要研究得细致入微。因此，我们也被逼着学会了如何认真仔细地思考并清晰准确地交流。如果我们做的事情不能满足他的标准，他就会把我们赶到一边，在我们的面前重做我们的工作。（我们把这样的经验教训称为“捣腾（Swooping）”，而且我们都很努力地工作以避免这种情况的发生。）

同样地，我们也学会了不理会由我们的供应商所选择的那些教程和参考资料。因为很明显，不管他们能否考虑得面面俱到，他们也可能和我们无法完全地沟通。可以这么说，我们将近 10 年的整个工作就是“捣腾”微软本身，即重新以短期课程、会议演讲、杂志文章和书籍等形式把微软的文档资料重新整理。我们称之为“微软编外人员就业行动（Microsoft Continuing Employment Act）”，这让我们的日子过得还挺滋润：微软吃肉，我们喝汤。

实际上，我们只需到处飞来飞去，说些诸如“记住调用 Release”、“避免往返过程”和“忽略聚合”的话就可以出色完成工作，因为我们有微软自己无法说清楚但对于开发人员很有意义的一系列明确的指导材料。但这也不是说在微软内部不存在能够清晰思考的人（Tony Williams 和 Crispin Goswell 就是两个我非常喜爱的人），不过那时候在初学者和能够阅读这样的高级著作的读者之间，还是存在着不可逾越的巨大鸿沟。

有了本书，开发 WPF 应用程序就可以得心应手了。Chris Anderson 是下一代 GUI 框架——Windows Presentation Foundation——的一名构架师，而这个 GUI 框架正是本书的主题。读者可能会认为，构架师的本职工作就是保证对问题的研究深入透彻，并正确地提出解决方法，以便其他人能够按部就班，做一些表面的事情就能让工作得以完成。而实际情况是，在本书中，Chris 从始至终都在指导开发人员的日常工作。Chris 对 WPF 内部的深刻理解可以为那些处于入门阶段的人们照亮学习之路，引导其深入了解他所创造（当然，我们也不能忘记这个创造还包括了超过 300 多个其他人的辛勤劳动）的这些基础功能。

本人也为其他出版社写了一本相同主题的书。我不能说本书是 WPF 学习中唯一需要的书籍（不然其他出版社会让我很“难堪”的），但是我可以肯定地说，这是一本需要时常翻阅的书。我也会拥有一本的。

Chris Sells sellsbrothers.com

2006 年 10 月

作者简介

Chris Anderson 是微软 Connected Systems 部门的一名构架师。Chris 主要关注的是用于实现下一代应用程序和服务的.NET 技术的设计和构架。

在微软的 10 年当中，他从事过 Visual Basic 6.0、Visual J++ 6.0、.NET Framework 1.0 和 1.1，以及最近的 Windows Presentation Foundation 的开发工作。Chris 的大部分职业生涯都涉及到界面技术——为 Visual Basic 构建控件、为 Visual J++开发 Windows Foundation Classes，以及在.NET Framework 中进行 Windows Forms 和 ASP.NET 开发。

2002 年，作为 Windows Presentation Foundation 的带头构架师，Chris 加入到了 Windows Client 团队中。凭借他的经验和实践，Chris 在无数个会议（PDC、TechEd、WinDev 和 DevCon 等）上做过演示和主题演讲。Chris 也在杂志和网站上发表了大量的文章。

Chris 的其他“嗜好”包括数码摄影、写作博客、视频游戏、潜水、汽艇和家庭影院，他的妻子和女儿都非常耐心地忍受他的这些“嗜好”。

这是 Chris Anderson 写作的第一本书。

前　言

在过去的9年时间里，我在微软参与了大量关于用户界面（UI）的项目。我在 Visual Basic 6.0、针对 Visual J++ 6.0 版本的 WFC（Windows Foundation Classes）、.NET Framework 中的 Windows Forms、一些从未被曝光过的内部项目和最终的 WPF（Windows Presentation Foundation）上都花过很多精力。

在 WPF 团队成立后，我于 2002 年秋天作为构架师加入该团队并开始了大约 18 个月的工作。从那时起，一直到 2005 年晚些时候，这个团队和技术的开发代号都叫 Avalon。在 2003 年早期，我获权协助重新设计这个平台，而且我们为在洛杉矶召开的 2003 年专业开发人员大会（Professional Developers Conference，PDC 2003）发布了一个技术预览版。WPF 是一个由超过 300 个人开发了近 5 年的产品。WPF 中的一些设计思想可以追溯到 1997 年的某些产品（例如，用于 Java 的 Application Foundation Classes 就是用于创建 WPF 组件的某些思想的起点）。

当我加入到 WPF 团队的时候，这个项目还处于研究阶段。这个项目包含了太多的想法，以致于不可能在一个单独的版本中全部发布。WPF 的首要目标是，使用一个融合了 Win32 和 Web 最佳特性的全新集成平台代替所有现存的创建客户端应用程序的基础结构。多么让人吃惊的雄心，它模糊了用户界面、文档和媒体之间的界限。在这几年中，我们做了一些痛苦的删减，添加了一些强大的特性，并从客户那里倾听了大量的反馈，但我们从未迷失正确的方向。

GUI 的历史简述

图形用户界面（GUI）源于 20 世纪 80 年代的 Xerox PARC 实验室。自那时起，微软、苹果及其他一些公司都为开发 GUI 应用程序创建了大量的平台。微软的 GUI 平台开始于 Windows 1.0，但一直到 1990 年发布 Windows 3.0 之前，它都未得到广泛的运用。构建 GUI 应用程序的基本编程模型由两个动态链接库（DLL）构成，即 User 和 GDI。1991 年微软发布了 Visual Basic 1.0，它构建于 User 和 GDI 之上，并提供了一个大量简化的编程模型。

Visual Basic 的 UI 模型在内部被称作“Ruby”的使用比原始的 Windows API 要简单得多。这种简单“激怒”了那些觉得编程应该困难复杂的人。然而早期版本的 Visual Basic 有很多限制，所以许多开发人员依然选择直接对 User 和 GDI 进行编程以构建“真正的”应用程序。随着时间的推移，一切都在改变。到微软的“世界”转到 32 位的时候，Windows 95 和 Visual Basic 4.0 发布了，VB 用户群获得了极大的动力，并被赋予了大量的平台特性。

与此同时，市场上发生的另外一个巨大变革是互联网。那时，微软正着手于一个在内部被称为“Forms3”的用于代替 Visual Basic UI 模型的项目。由于多方面的原因，微软决定使用这个模型作为浏览器应用程序平台的基础。这个引擎在内部被重命名为“Trident”，现在它已经作为 MSHTML.dll 函数库在 Windows 中出现。Trident 促进了具有丰富文本布局、标记和脚本支持的 HTML 特定引擎在最近几年中的发展。

几乎同时，另外一种现象也出现在人们的视野中，即托管代码（Managed Code）。Visual Basic 已经在托管环境中运行很长时间了（其他的一些语言也是如此），但是，由 Sun Microsystems 于 1994 年引入的 Java 第一次让大量的开发人员接受了虚拟机（Virtual Machine）的概念。在接下来的几年中，托管代码在市场上越来越有影响力。2002 年，微软发布了自己的通用托管代码平台：.NET Framework。包含在.NET Framework 中的 Windows Forms 是一个针对 User32 和 GDI+（GDI32 的升级版本）的托管代码 API。Windows Forms 倾向于代替 Visual Basic 中的旧的“Ruby”窗体开发包。

进入到新千年后，微软拥有 4 个主要的 UI 平台：User32/GDI32、“Ruby”、“Trident”和 Windows Forms。这些技术解决了不同方面的问题，具有不同的编程模型，且被不同的用户群所使用。图形系统也被涉及。1995 年，微软引入了 DirectX 技术，让程序员可以更深入地控制硬件的图形系统。可是，这 4 项主要的 UI 技术都没有以一种更有意义的方式利用这项新近出现的强大技术。

在这里，还有一个现实问题有待解决。客户要求在他们的应用程序中能提供类似现代视频游戏的丰富体验和视频播放能力。媒体、动画和丰富图形应该无处不在。他们希望富文本能被支持，因为几乎每个应用程序都要显示某种类型的文本或文档。他们希望有丰富的用于创建应用程序的 Widgets、Buttons、Trees、Lists 和 Text Editors（文本编辑器）——所有这些在创建基本应用程序的时候都是必需的。

这 4 个主要平台可以满足很大一部分客户需求，但它们都是彼此孤立的，要混合和配合这些平台中的各个部分是困难且容易出错的。从一个纯粹自私的角度看，微软的管理层已经疲于继续投资这 4 个团队以开发大致重复的技术。

2001 年，微软成立了一个新团队，这个新团队有一个听起来简单的使命：创建一个统一的界面呈现平台，以最终代替 User32/GDI32、“Ruby”、“Trident”和 Windows Forms，并能提供新的开发方案以满足客户对呈现技术的需求。这个团队的大量成员来自现有的呈现平台

团队，他们的目标是开发一个最佳的同类平台，这可真是一次飞跃性的进步。

于是，Aralon 团队成立了。在 2003 年举办的 PDC 期间，微软发布了 Aralon（当时的代号）。后来，该项目被命名为 WPF（Windows Presentation Foundation）。

WPF 原则

尽管创建 WPF 花费了很多时间，但是就该项目的整个生命周期而言，有几条指导性的原则却始终保持不变。

创建一个用于丰富呈现的平台

在描述一项新技术的时候，“丰富”大概是最常用的词汇。然而，我实在想不出更好的术语来传达 WPF 背后的原则。我们的目标是根据所有现存的呈现技术来创建一个功能特性的超集——从诸如矢量图形、渐变效果和位图效果等基本功能，到诸如 3D、动画、媒体和印刷体等高级功能。这个原则的其他关键部分即是“平台”。其目标是不仅仅创建一个用于丰富内容的运行时“播放器”，还要创建一个应用程序平台，让人们能够使用它创建具有良好伸缩性的应用程序，甚至能通过扩展平台来完成我们从未想象过的新功能。

创建一个可编程的平台

在早期，WPF 团队就认为“标记”（声明式）和“代码”（命令式）的编程模型都是平台所需要的。那时，我们清楚地看到，开发人员已经接受了新的托管代码环境。很快地，可编程平台的原则变成了托管编程模型原则。其目标是让托管代码成为系统的原生编程模型，而不是一个附加层。

创建一个声明式的平台

从客户和软件开发者的角度看，显然业界正使用越来越多的声明式编程模型。我们确信，要让 WPF 获得成功，需要丰富、一致和完整的标记编程模型。另外，业界正在发生的一切已经说明，XML 正在成为数据交换事实上的标准，所以我们决定创建一个 XML 编程模型，这就是 XAML（扩展应用程序标记语言，Extensible Application Markup Language）。

集成 UI、文档和媒体

创建应用程序的客户面对的最大问题可能是，功能块被分离到独立的“孤岛”上。这就是说，一个平台用于创建用户界面，另外一个用于创建文档，并且还需要一个宿主平台来创建媒体，这依赖于他们的介质（3D、2D、视频、动画等）。在着手构建一个全新的呈现系统之前，我们就设定了一个非实现不可的目标：集成 UI、文档和媒体将是整个团队最优先考虑的事情。

融合 Web 和 Windows 的优点

其目标是从最近 20 年的 Windows 开发中和最近 10 年的 Web 开发中吸取最佳的特性，

并创建一个全新的平台。Web 提供了强大而简单的标记模型、开发模型、应用程序的通用结构和丰富的服务器连通性。Windows 提供了丰富的客户端模型、简单的编程模型、应用程序外观的控制机制和丰富的网络服务。这项任务是要模糊 Web 应用程序和 Windows 应用程序之间的界限。

集成开发人员和设计师

由于应用程序变得更加图形化且更加迎合用户的体验，因此一个崭新的开发团体必须被集成到开发过程中。媒体公司（印刷业、在线媒体、电视台等）早就知道，一些设计师必须参与进来，以为客户创建更好的用户体验。现在我们看到，对于软件也有着同样的需求。从历史来看，设计师使用的工具和软件开发过程是完全没有关联的：设计师使用类似 Adobe Photoshop 或 Adobe Illustrator 这样的工具来创建漂亮的设计效果，而当开发人员实现这些设计效果时却受到了阻碍。创建一个能天然地支持设计师所要求的特性的统一系统，以及利用能让工具之间进行无缝协作的标记格式（XAML），这就是这个原则的两个结果。

关于本书

现在，关于 WPF 的书有很多，将来还会更多。当第一次考虑要写本书时，我打算在书中提供一些独一无二的内容。本书专为应用程序开发人员而写，其目标是成为一本覆盖 WPF 大部分内容的概念参考书。

我谨慎地在前面的叙述中选择每个词汇。

本书是关于应用程序的。实际上存在两种类型的软件：设计用于和人们交互的软件，以及设计用于和软件交互的软件。我使用“应用程序”这个术语来表明这里的“软件”主要指和人交互的软件。从根本上说，WPF 是关于和人交互的技术。

本书是针对开发人员的。我打算介绍这个平台中以代码为核心的部分。首先，我是一个开发人员。作为一个构架师工作于 WPF 团队时，我总是把外部的开发人员认为是首要的客户。本书关注的主题主要是针对应用程序开发人员的。虽然一个控件开发人员也可以在本书中发现很多有用的信息，但本书的目标不是介绍如何创建自定义控件。

本书是关于概念的，而不仅仅是关于 API。如果读者需要一个 API 的参考信息，可使用 Google 或 MSN 进行搜索，也可以浏览 MSDN 文档。我打算上升到一个抽象的高度来介绍这个平台是如何设计的以及为什么这样设计，并演示这个平台的一些 API 如何协同工作来为开发人员提供更多的价值。本书是参考资料；它以技术主题进行组织，以便读者能跳到某章的后面或跳到某章的前面来寻找相关答案。读者不需要从头至尾地阅读本书，就可以从中得到收获。

本书覆盖了 WPF 的大部分内容，当然没有包含全部内容。当我开始写本书的时候，Chris Sells 给了一个重要的建议："你略去的和你包含的一样重要"。因为 WPF 是一个浩瀚的平台，我必须略去一些部分，才能呈现出重要的部分。本书讲述了我认为在研究这个平台的过程中最有意义的内容。

本书的目标是展现 WPF 核心概念的图景，说明他们相互之间是如何关联的，以及解释是什么促成了这样的设计。我希望读者能够摆脱本书对 WPF 的理解，自己能够更深入地研究这个平台。

阅读要求

在阅读本书之前，读者应该对.NET 比较熟悉。读者不需要成为一个专家，但应该熟悉关于类、方法和事件的基本知识。本书在示例中只使用了 C#代码。WPF 对任何.NET 语言都一视同仁。不过，C#是我在开发中使用的主要语言。

本书结构

本书分为 8 章和 1 个包含 3 个部分的附录。本书的目标是以尽可能少的章节来讲述 WPF 平台。

- **导言**（第 1 章）简要地介绍这个平台，并解释 WPF 中的 7 个主要组件是如何协作的。作为使用 WPF 创建应用程序的一个快速入门，本章也演示了如何使用 SDK 工具，以及如何在文档中找寻帮助内容。
- **应用程序**（第 2 章）介绍使用 WPF 构建的应用程序的结构，以及被应用程序所使用的一些应用服务和顶级对象。
- **控件**（第 3 章）既介绍 WPF 控件中的主要设计模式，也介绍 WPF 中的主要控件类别。控件是 WPF 的一些基础用户界面构建块；如果读者只想阅读本书的一章，那么本章最适合不过。
- **布局**（第 4 章）介绍布局系统的设计，以及 WPF 中内置的 6 个现成布局板。
- **可视化效果**（第 5 章），介绍"巨大平面区域"即 WPF 可视化系统。本章涉及的内容包括印刷版面式样、2D 和 3D 图形、动画、视频和音频。
- **数据**（第 6 章）介绍数据源、数据绑定、资源和数据传递操作的基础知识。
- **动作**（第 7 章）介绍如何让事件、命令和触发器在应用程序里执行。

- **样式**（第 8 章）介绍 WPF 中的样式系统。样式功能通过让 UI 的可视化外观和编程结构松散耦合，以使设计师和开发人员明确地分离。
- **基础服务**（附录）深入研究 WPF 中的一些低级服务。涉及到的主题包括线程模型、属性和事件系统、输入、排版和打印。

致谢

对于我来说，本书是一个宏大的工程。我之前写过文章、演示稿和白皮书，但这些都没能让我为如此繁重的工作——把 WPF 平台的很多内容压缩到一本相对简短的书中——做好准备。

我要把本书献给我的妻子——Megan。她自始至终支持这个项目（甚至在很多次度假的时候，我都带着笔记本）和我做的其他所有事情。

在完成本书（及本项目）的过程中，整个 Avalon 团队也给予了我巨大的帮助。我的经理 Ian Ellison-Taylor 一直支持我在这个项目中的工作。Sam Bent、Jeff Bogdan、Vivek Dalvi、Namita Gupta、Mike Hillberg、Robert Ingebretsen、David Jenni、Lauren Lavoie、Ashraf Michail、Kevin Moore、Greg Schechter……帮助过我的团队成员太多了，这里很难全部列出。我和团队中的每个人都合作得非常愉快。

我很感激 Don Box 一直鼓励我写作本书，还有 Chris Sells 在我们讨论这本书的过程中给予了很多高见。

我的责任编辑 Michael Weinhardt 为本书的质量做了大量的工作。Michael 对本书的每一章读了又读，编辑了又编辑。他督促我不要接受任何不完善的地方。本书中的所有错误和失败之处纯粹是我的失误。

Joan Murray、Karen Gettman、Julie Nahil 和 Addison-Wesley 的全体工作人员都为本书做了大量工作。Stephanie Hiebert——我的副本编辑——在我糟糕的拼写、语法和文风上花了大量时间，把我凌乱的词句修改成了“真正的英语”。

最后，我要感谢本书的技术审校者。Erick Ellis、Joe Flanigan、Jessica Fosler、Christophe Nasarre、Nick Paldino、Chris Sells 和其他很多人都提供了非常好的反馈。Jessica 给了我一些很有深度和非常有建设性的批评意见，而且我已经听取了这些意见。

我敢肯定，我还遗忘了很多帮助过我的人，在此表示歉意。

Chris Anderson

simplegeek.com

2006 年 11 月

目　录