谨以此书，呈现一种掌握WPF的轻松方式，分享一个微软技术粉丝的治学精神

深入浅出WPF

深入之美 精辟分析WPF源代码，洞察功能背后隐藏的深刻设计理念
浅出之美 最浅显的生活案例，融化最抽象的高级概念
分离之美 掌握UI与逻辑如何真正分离，享受变与不变的快感
自然之美 全新的数据驱动UI理念，让数据重归核心地位
感观之美 强大的图形引擎，实现绝对震撼的UI视觉

刘铁猛 著

中国水利水电出版社
www.waterpub.com.cn

谨以此书，呈现一种掌握WPF的
轻松方式，分享一个微软技术粉丝的治学精神

深入浅出WPF

深入之美
浅出之美
分离之美
自然之美
感观之美

精辟分析WPF源代码，洞察功能背后隐藏的深刻设计理念
最浅显的生活案例，融化最抽象的高级概念
掌握UI与逻辑如何真正分离，享受变与不变的快感
全新的数据驱动UI理念，让数据重归核心地位
强大的图形引擎，实现绝对震撼的UI视觉

刘铁猛 著

中国水利水电出版社
www.waterpub.com.cn

内 容 提 要

WPF 是微软新一代开发技术，涵盖了桌面应用程序开发、网络应用程序开发和移动应用程序开发，是微软开发技术未来十年的主要方向。

本书的内容分为两大部分。第一部分是学习 WPF 开发的基础知识，包括 XAML 语言的详细剖析、WPF 控件的使用、用户界面布局的介绍。第二部分是作为优秀 WPF 程序员所应掌握的知识，包括依赖对象和数据关联、路由事件与命令、数据模板与控件模板、绘图与动画等。

本书作者具有多年 WPF 开发经验，历经多个大型项目，现任微软（美国）下载中心项目组高级开发工程师。本书是作者多年来学习和使用 WPF 的经验总结。

本书包含了众多 WPF 面试点，作者凭借书中的知识顺利通过微软（美国）的面试。

图书在版编目（CIP）数据

深入浅出WPF / 刘铁猛著. -- 北京 : 中国水利水电出版社, 2010.7
ISBN 978-7-5084-7635-3

Ⅰ. ①深… Ⅱ. ①刘… Ⅲ. ①窗口软件，Windows Vista－用户界面－程序设计 Ⅳ. ①TP316.7

中国版本图书馆CIP数据核字(2010)第120800号

策划编辑：周春元　　责任编辑：杨元泓　　加工编辑：胡海家　　封面设计：李　佳

书　　名	深入浅出 WPF
作　　者	刘铁猛　著
出版发行	中国水利水电出版社 （北京市海淀区玉渊潭南路 1 号 D 座　100038） 网址：www.waterpub.com.cn E-mail：mchannel@263.net（万水） sales@waterpub.com.cn 电话：（010）68367658（营销中心）、82562819（万水）
经　　售	全国各地新华书店和相关出版物销售网点
排　　版	北京万水电子信息有限公司
印　　刷	北京蓝空印刷厂
规　　格	184mm×240mm　16 开本　19 印张　437 千字
版　　次	2010 年 7 月第 1 版　2010 年 7 月第 1 次印刷
印　　数	0001—4000 册
定　　价	45.00 元

I

写作缘起

本书的写作缘起几年前我学习 WPF。因为我是从 Windows Forms 开发转来做 WPF 开发的，学习过程中遇到很多新概念、新特性，其中包括 Data Binding、路由事件、命令、各种模板等。我的工作风格是对于每个新知识，一定先把它理解透彻、搞明白再应用于项目中，不然总感觉使用起来不放心，于是就对照已有的英文书籍和 MSDN 逐一研究这些知识点。每有所得，都喜欢写成博客发表在网上，一来供大家学习参考，二来做一个积累、防止以后遗忘。博客发表之后收到很多读者的反馈和鼓励，大家希望我能把这些文章编撰成册、形成一本学习教材，于是我下决心开始写这本书。这本书的名字也就随了系列博客文章的名字——《深入浅出 WPF》。

之所以叫“深入浅出”，原因有两个。名为“深入”，是想把 WPF 也诠释一番，所以书中的每个例子都有可供剖析的实例，对于一些重要概念，我通过分析 WPF 的源代码给予阐述（.NET Framework 的部分源代码是向开发人员开放的，其中就包含 WPF 的源代码）。名为“浅出”，是因为几乎每个概念我都会用生活中浅显易懂的例子进行类比，让读者可以轻松理解，降低学习抽象知识的痛苦。

为本书起这个名字，也是出于我对《深入浅出 MFC》这本书的景仰之情。我刚刚开始学编程的时候正是 MFC 流行的年代，《深入浅出 MFC》这本书给我的学习风格打下了深深的烙印。其中对我影响最深刻的，一个是它对 MFC 源码的分析，另一个是“勿在浮沙筑高台”、凡事必究其理的探索精神。在后来的近十年工作中，分析和学习微软开发框架的源码成为我工作的方法论。本书中包含了一些对 WPF 源码的分析，帮助大家对 WPF 有个透彻的理解。我以《深入浅出 MFC》一书为准绳和鞭策自己的力量，希望能为大家奉上一本有用的好书。

写博客容易，写书难。写博客，内容上可以不那么连贯、不太严谨，写书就不一样了，要求每个知识点都要仔细琢磨、谨慎下笔，经常是写了满满一篇之后感觉不满意又删掉重来，直到我认为初级读者也能顺畅理解为止。多少个不眠之夜就是在这种字斟句酌中转瞬即逝，一年下来，头上也冒出了很多白发。我想，既然写书，那就要把自己的心血奉献给读者，这样才对得起读者也对得起自己。

本书并不是一本大而全的 WPF 宝典，而是 WPF 在实际工作中用到最多的部分。所以在“轻松幽默、深入浅出”的风格基础上，本书力求实用。写书的过程其实也是对 WPF 进行深耕的过程，本书写作过半时，我偶然获得一个机会可以参加微软的一个开发项目，面试我的是美国微软的一位高级项目经理（现在是我的老板），面试的内容就是 WPF 开发。我基本上都是用书中的原话作答，十分顺利——我获得了来美国工作的机

会，目前负责微软下载中心管理工具的开发。我想，这也算是对本书内容的一次检验，衷心希望大家在学习完这本书中的内容后也能在自己的职业发展上获得进步。

毕竟我的水平有限，尽管下力气去写但还是感觉很粗浅；有些知识超出微软官方文档的覆盖，我也融入一点自己的判断，对 WPF 源码的阅读也是在探索中前行，所以，书中疏漏之处再所难免。希望大家能够多多给予宽容并提出宝贵的建议。我将在本书的后续版本中不断丰富内容、修改错误，让这本书成为一本“活书”、一直为大家服务下去。本书的纠错及更正将发布在 http://www.cnblogs.com/prism 。我的 MSN 是 wpfgeek@live.com，期待与热爱 WPF 技术的朋友共同学习和探讨。

II

WPF 之 What & Why

自古以来，生产工具就代表着生产力的先进程度——生产力的发展要求人们不断研发出新的生产工具，新生产工具的诞生又使生产效率出现飞跃。作为劳动生产的一种，计算机软件开发也需要工具，随着程序员们手中的工具越来越强大，软件开发的效率和质量也越来越高。善于学习和掌握新工具、新技术的程序员们也总是能得到更多的实惠。

微软 Windows 操作系统成功推出已有十多年，在 Windows 系统平台上从事图形用户界面（Graphic User Interface，GUI）程序开发的程序员数不胜数。GUI 程序员们手中的开发工具历经了 Win32 API→MFC（及同类产品）→ActiveX/COM/Visual Basic→Windows Forms 的变迁，每一次变迁都使开发效率和质量产生飞跃。从 2007 年开始，微软推出了它的新一代 GUI 开发工具 Windows Presentation Foundation（直译为 Windows 表示基础，WPF），并且把 WPF 定为未来十年 Windows 平台 GUI 开发的主要技术。时至今日，不但 Windows Vista、Windows 7、Windows Server 2008、Windows Server 2008 R2 等系统已经无缝集成了 WPF，连 Visual Studio 2010 等重要产品业已使用 WPF 进行开发。可见微软在 WPF 技术方面的务实精神与决心。

什么是 WPF

WPF 是 Windows Presentation Foundation 的简称，顾名思义是专门用来编写程序表示层的技术和工具。

WPF 是做什么用的呢？让我们从分析一个客户的需求开始，解答这个问题。经常会有一些朋友找我写项目，有一次，一家医疗单位的技术主管找到我说：“你能不能用 WPF 为我们开发一套管理系统呀？”其实，这就是一个对 WPF 的典型误解。误解在何处呢？主要是没有弄清 WPF 的功用。当今的程序，除了一些非常小巧的实用工具外，大部分程序都是多层架构的程序。一提到多层架构，一般就至少包含 3 层：数据层、业务逻辑层和表示层（它们的关系如图 1 所示）。

这 3 层的功能大致如下：

- 数据层：用于存储数据，多由数据库构成，有时候也用数据文件能辅助存储数据。比如医院的药品列表、人员列表、病例列表等都存储在这一层。
- 业务逻辑层：用于根据需求使用计算机程序表达现实的业务逻辑。比如哪些医生可以给哪些病人看

病，从挂号到取药都有什么流程，从住院到出院有哪些流程，都可以由这层来实现。这一层一般会通过一组服务（Service）向表示层公开自己的各个功能。因为这一层需要与数据层进行交互，所以经常会划分出一个名为“数据访问层”（Data Access Layer，DAL）的子层专门负责数据的存取。

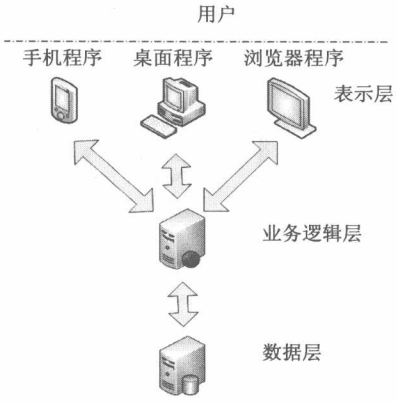

图 1

- 表示层：负责把数据和流程展示给用户看。对于同一组来自业务逻辑层的数据，我们可以选择多种表达方式。比如对于同一张药品单，如果想以短信的形式发送给药房，可以以一串字符的形式来表达；如果客户想打印药品单的详细内容，可以以表格的形式来表达；如果客户想直观地看到每种药品占总价格的比例，我们可以使用饼图来表达。除了用于表示数据，表示层还负责展示流程、响应用户操作等。而且，表示层程序并不拘泥于桌面程序，很多表示层程序都运行在手机或浏览器里。表示层程序也常被称为客户端程序。

WPF 的功能就是用来编写应用程序的表示层，至于业务逻辑层和数据层的开发也有专门的新技术。比如业务逻辑层的新技术是 WCF（Windows Communication Foundation）和 WF（Windows Workflow Foundation）。微软平台上用于开发表示层的技术不算少，包括 WPF、Windows Forms、ASP.NET、Silverlight 等。换句话说，无论使用哪种技术作为表示层技术，程序的逻辑层和数据层都是相同的。所以“使用 WPF 开发管理系统”这个提法是不对的。

WPF 与 Silverlight 的关系

目前，.NET 开发人员学习 WPF 的回报是相当高的，原因是几乎整个微软新一代开发框架都能看到 WPF 的影子。微软的新一代开发技术框架包括 Windows Presentation Foundation（WPF）、Windows Communication Foundation（WCF）和 Windows Workflow Foundation（WF，据说因为与 World Wildlife Fund 的缩写 WWF 冲突了，所以去掉了一个 W）。本书无疑是讲 WPF，而 WCF 的用途是编写分布式应用程序的业务逻辑层，并以网络服务的形式暴露给客户端的服务消费者，基于 WCF 和 Entity Framework 的 WCF Data Service 和 WCF RIA Service 是微软迄今最佳的数据访问层，而这一数据访问层的最佳消费者就是 WPF 和 Silverlight。所以，学习 WPF 技术可以为 WCF 的学习锦上添花。WF 的主要作用是设计工作流，而设计工作流的编程语言正是 WPF

中的界面设计语言——XAML，也就是说，学习完 WPF，WF 也会了一小半。

如果说学会 WPF 后 WF 算是会了一小半，那么学习完 WPF 后，Silverlight 可以算是会了 80%，为什么这么说呢？因为微软原本就定义 Silverlight 是 WPF 的一个子集、是 WPF 的“网络版”（Silverlight 的开发代号是 WPF/E，意为 WPF 简化版）。为了让 WPF 在浏览器里跑起来，微软所做的事情就是在技术理念不变的情况下对 WPF 进行“瘦身”——去掉一些不常用的功能、简化一些功能的实现，对多组实现同一目的的类库进行删减、只保留一组，再添加一些网络通信的功能。通过下表，我们就能看到 Silverlight 与 WPF 的技术重叠率之高：

技术项目	在 WPF 中	在 Silverlight 中
XAML 语言	完整	完整
控件	完整	完整
布局	完整	完整
Binding	完整	基本完整
依赖属性	完整	基本完整
路由事件	完整	简化
命令	完整	无
资源	完整	完整
控件模板	完整	基本完整
数据模板	完整	基本完整
绘图	完整	完整
2D/3D 动画	完整	简化

如今 Silverlight 炙手可热的另一个原因是微软新一代手机平台 Windows Phone 7 也采用它来作为开发平台（此前的 Windows Mobile 系统使用的是简化版的 Windows Forms 开发平台）。Windows Phone 7 中运行的 Silverlight 与浏览器中运行的 Silverlight 别无二致，因此学习完 WPF 之后连手机平台上的程序也会写了。

所以说学习 WPF 是“一本万利”的投资，一点也不过分。

为什么要学习 WPF

有的朋友会问：既然已经有这么多表示层技术，为什么还要推出 WPF 技术呢？我们花精力学习 WPF 技术有什么收益和好处呢？这个问题可以从两方面来回答。

首先，只要开发表示层程序就不可避免地要与 4 种功能性代码打交道，它们分别是：

- 数据模型：现实世界中事物和逻辑的抽象。
- 业务逻辑：数据模型之间的关系与交互。
- 用户界面：由控件构成的、与用户进行交互的界面，用于把数据展示给用户并响应用户的输入。

- 界面逻辑：控件与控制之间的关系与交互。

这 4 种代码的关系如图 2 所示。

图 2

在保持代码可维护性的前提下，如何让数据能够顺畅地到达界面并灵活显示，同时方便地接收用户的操作历来都是表示层开发的核心问题。为此，人们研究出了各种各样的设计模式，其中有经久不衰的 MVC（Model-View-Controler）模式、MVP（Model-View-Presenter）模式等。在 WPF 出现之前，Windows Forms、ASP.NET（Web Forms）等技术均使用“事件驱动”理念，这种由“事件→订阅→事件处理器”关系交织在一起构成的程序，尽管可以使用 MVC、MVP 等设计模式，但一不小心就会使界面逻辑和业务逻辑纠缠在一起，造成代码变得复杂难懂、bug 难以排除。而 WPF 技术则是微软在开发理念上的一次升级——由“事件驱动”变为“数据驱动”。

事件驱动时代，用户每进行一个操作用会激发程序发生一个事件，事件发生后，用于响应事件的事件处理器就会执行。事件处理器是一个方法（函数），在这个方法中，程序员可以处理数据或调用别的方法，这样，程序就在事件的驱动下向前执行了。可见，事件驱动时代的数据是静态的、被动的；界面控件是主动的、界面逻辑与业务逻辑之间的桥梁是事件。而数据驱动正好相反，当数据发生变化时，会主动通知界面控件、推动控件展示最新的数据；同时，用户对控件的操作会直接送达数据，就好像控件是“透明”的。可见，在数据驱动理念中，数据占据主动地位、控件和控件事件被弱化（控件事件一般只参与界面逻辑，不再染指业务逻辑，使程序复杂度得到有效控制）。WPF 中，数据与控件的关系就是哲学中内容与形式的关系——内容决定形式所以数据驱动界面，这非常符合哲学原理。数据与界面之间的桥梁是数据关联（Data Binding），通过这个桥梁，数据可以流向界面，再从界面流回数据源。

简而言之，WPF 的开发理念更符合自然哲学的思想（除了 Data Binding 之外，还有 Data Template 和 Control Template 等，本书都将着力描述）。使用 WPF 进行开发较之 Windows Forms 开发要简单，程序更加简洁清晰。

其次，微软已经把 WPF 的理念扩展到了几乎全部开发平台，包括桌面平台、浏览器平台和手机平台。WPF 的完整版可用于在 Windows 平台上开发桌面应用程序（这些桌面应用程序也可以运行在浏览器中）；WPF

的“简化版”，也就是 Silverlight，不但可以用于编写运行在浏览器中富客户端程序（Rich Internet Application），也可以用于编写运行于微软最新手机平台 Windows Phone 7 中的程序。所有这些程序的开发理念都是一样的，仅在类库方面有细微的差别，也就是说，学会 WPF 开发，Silverlight 开发和 Windows Phone 7 开发均可触类旁通。所以学习 WPF 的发展前景非常好、回报很大，投入些精力是非常值得的。

最后，为大家提供一个微软官方集动画和效果为一体的实例，photoSuru。下载地址是：http://windowsclient.net/appfeeds/SubscriptionCenter/Gallery/photosuru.aspx。

III

致　谢

本书的出版要感谢中国水利水电出版社万水分社的周春元编辑。在整个成书过程中，春元兄一直保持着极大的耐心，总是给予我真诚的支持和鼓励。记得有一次，我对自己的技术水平不够自信，春元兄问我："你为什么要写这本书？"我说是想为学习 WPF 的朋友做点事情、对中国的 IT 行业做点推动。春元兄说："是啊！所以只要你尽心尽力、把真知奉献给读者，就已经是在帮助大家学习、实现自己的理想了。至于书是不是好卖，这个不用担心，只要是好书，就算赔着钱我们也要出、也要卖。"当我思路枯竭的时候，春元兄从来没有以稿期为由催过我，只是鼓励我让我静下心来慢慢写；春元兄也从来没有要求我为迎合销售而对书稿做丝毫改动，是春元兄帮助我保证了这本书是一本没有铜臭、不掺杂名利的书。

最让我感动的是，为了保证本书的质量，春元兄作为一个非计算机编程人士竟然通读了整本书、几乎弄懂了 WPF 的所有概念，最后提出了很多非常宝贵的意见和见解，有些意见甚至精确到一个词或一个字母的使用。

成书过程中与我直接沟通的是春元兄，但站在春元兄背后的是一个完整的编辑团队——他们默默无闻地工作着，保证书籍以最快的速度、最优雅的阅读体验、最精美的印刷与读者们见面。在此，向杨元泓、胡海家、陈洁等编辑朋友致以真诚的感谢。

我之所以能成为程序员、成为技术作者，首先要感谢我的父母，是他们培养我勤于动手、乐于思考、善于表达。刘晓林先生是我的计算机启蒙老师，如果没有他带我从 DOS 学起，很难想象我会成为一名程序员。初中的李全兴老师和高中的郭惠清老师是两位对我影响最大的语文老师，没有二位老师在写作方面的指导和帮助，别说是写书，恐怕写作文都成问题。一路走来，有良师，还有益友——刘扬、张博、谢志威、常诚等，他们不但是在计算机学习方面对我帮助最大的朋友，也是我的知心朋友，在此，对你们表示衷心的感谢，愿友谊长青！

在成长过程中，无数的朋友帮助过我、引导过我，众多的领导关心我、鼓励我，在此一并表示感谢。如今的我已经懂得在平和中稳步前行，而在此之前，很多前进的动力则来源于挫折和挑战，对曾经的挫折与挑战，我也表示深深的感谢——感谢你们让我思考、成长和成熟。

刘铁猛

2010 年 6 月 18 日

于 Redmond，Microsoft Building Mil-E

IV

目　录

第二部分游历 WPF 内部世界

第一部分

深入浅出话 XAML

1 XAML 概览

1.1 XAML 是什么

自人类社会诞生，社会分工就在不断地进行着。从原始社会畜牧业与农业分离到当今数以万计行业的存在，无不是社会分工的杰作。社会分工的意义在于它能使从事固定工作的人群更加专业化，并通过合作的形式提高生产效率。换句话说，在合作不是问题的情况下，若干群专业人士配合工作要比同等数量的一群“大而全”人士的工作效率高。

这种分工与合作的关系不仅存在于行业之间，也存在于行业内部。软件开发中最典型的分工合作就是设计师（Designer）与程序员（Programmer）之间的协作。在 WPF 出现之前，协作一般是这样展开的：

（1）需求分析结束后，程序员对照需求设计一个用户界面（User Interface，UI）的草图，然后把精力主要放在实现软件的功能上。

（2）与此同时，设计师们对照需求、考虑用户的使用体验（User Experience，UX）、使用专门的设计工具（比如 Photoshop）设计出优美而实用的 UI。

（3）最后，程序员按照设计师绘制的效果图，使用编程语言实现软件的 UI。

经验告诉我们，即便是优秀的设计师团队和优秀的开发团队合作，花费在沟通和最终整合上的精力也是巨大的。经常出现的问题有：

- 设计师的设计跟不上程序逻辑的变化。
- 程序员未能完全实现设计师提供的效果图。
- 效果图与程序功能不能完全匹配。
- 从效果图到软件 UI 的转化耗费很多时间。

这些并不是谁对谁错的问题——只要存在分工，合作的成本就不可能为零。问题的核心在于，设计师与程序员的合作是“串行”的，即先由设计师完成效果图、再由程序员通过编程实现。如果

设计师能与程序员“并行”工作并直接参与到程序的开发中来，上述问题就解决了。

解决方案是什么呢？是让设计师们使用编程语言来设计 UI 效果图，还是让程序员们使用 Photoshop 来开发程序？显然都行不通。

网络程序开发团队的经验倒是很值得借鉴：草图产生后，设计师们可以使用 HTML、CSS、JavaScript 直接生成 UI，程序员则在这个 UI 产生的同时实现它背后的功能逻辑。在这个并行的合作中，设计师们可以使用 Dreamweaver 等设计工具，程序员使用 Visual Studio 来进行后台编程。有经验的设计师和程序员往往还具备互换工具的能力，使得他们能基于 HTML+CSS+JavaScript 这个平台进行有效的沟通。

为了把这种开发模式从网络开发移植到桌面开发和富媒体网络程序的开发上，微软创造了一种新的开发语言——XAML（读作 zaml）。XAML 的全称是 Extensible Application Markup Language，即可扩展应用程序标记语言。它在桌面开发及富媒体网络程序的开发中扮演了 HTML+CSS+JavaScript 的角色、成为设计师与程序员之间沟通的枢纽。

现在，设计师和程序员们一起工作、共同维护软件的版本，只是他们使用的工具不同——设计师们使用 Blend（微软 Expression 设计工具套件中的一个）来设计 UI，程序员则使用 Visual Studio 开发后台逻辑代码。Blend 使用起来很像 Photoshop 等设计工具，因此可以最大限度地发挥出设计师的特长。使用它，设计师不但可以制作出绚丽多彩的静态 UI，还可以让 UI 包含动画——虽然程序员们也能做出这些东西，但从专业性、时间开销以及技术要求上显然是划不来的。更重要的是，这些绚丽的 UI 和动画都会以 XAML 的形式直接保存进项目，无需转化就可以直接编译，节省了大量的时间和成本。

注意

下次，当你在面试被问到“什么是 XAML”时，你可以回答：XAML 是 WPF 技术中专门用于设计 UI 的语言。

1.2 XAML 的优点

前面一节已经向我们透露了 XAML 的几个优点：

- XAML 可以设计出专业的 UI 和动画——好用。
- XAML 不需要专业的编程知识，它简单易懂、结构清晰——易学。
- XAML 使设计师能直接参与软件开发，随时沟通、无需二次转化——高效。

然而，XAML 这位翩翩君子的才华可远不止这些。

自从应用程序从控制台界面（Console User Interface，CUI）升级为图形用户界面（Graphic User Interface，GUI）后，程序员们就一直追求将视图（View，也就是 UI）与逻辑代码的分离。以往的开发模式中，程序员很难保证用来实现 UI 的代码完全不与用来实现程序逻辑的代码纠缠在一起。UI 代码与逻辑代码纠缠在一起称为 UI 与逻辑的紧耦合，它往往带来以下的后果：

- 无论是软件的功能还是 UI 设计有所变化或者是出了 bug，都将导致大量代码的修改。