

# XAML

## Developer Reference



Mamta Dalal  
Ashish Ghoda

# XAML Developer Reference

## Your expert guide to designing and building dynamic user interfaces

Sharpen your application design and development skills using XAML—the declarative markup language used in Microsoft® Silverlight®, Windows® Presentation Foundation (WPF), and the Windows 8 Runtime APIs. Led by two XAML experts, you'll learn practical ways to build rich, interactive user interfaces with data integration capabilities and support for multimedia, graphics, and animation. This hands-on guide is ideal for Microsoft .NET developers and web designers alike.

### Discover how to:

- Control UI behavior and implement business logic with code-behind solutions
- Manage UI element positioning with the XAML layout system
- Use templates to customize UI elements—without affecting their functionality
- Apply different types of property and event systems in WPF and Silverlight
- Bind various kinds of data to your UI, and display them in the format you want
- Implement 2D and 3D vector graphics and animations
- Reuse control styles and properties to maintain consistency throughout your application



### Get code and project samples on the web

Ready to download at  
<http://go.microsoft.com/FWLink/?Linkid=233593>

For **system requirements**, see the Introduction.

[microsoft.com/mspress](http://microsoft.com/mspress)

ISBN: 978-0-7356-5896-7



**U.S.A. \$39.99**

**Canada \$41.99**

[Recommended]

Programming/Windows/  
Microsoft Visual Studio



### About the Authors

**Mamta Dalal** has more than 10 years of experience developing Windows and web applications. She is an active contributor to the .NET community and has written several articles on C#, .NET, Silverlight, and WPF.

**Ashish Ghoda**, founder and president of a technology consulting firm, is an accomplished author with more than 15 years of experience in enterprise architecture, application development, and technical and financial management.

### RESOURCE ROADMAP

#### Developer Step by Step

- Hands-on tutorial covering fundamental techniques and features
- Practice exercises
- Prepares and informs new-to-topic programmers



#### Developer Reference

- Expert coverage of core topics
- Extensive, pragmatic coding examples
- Builds professional-level proficiency with a Microsoft technology



#### Focused Topics

- Deep coverage of advanced techniques and capabilities
- Extensive, adaptable coding examples
- Promotes full mastery of a Microsoft technology



See inside cover

 **Windows®**

**Microsoft®**

# XAML Developer Reference

Mamta Dalal  
Ashish Ghoda

Copyright © 2011 by Mamta Dalal and Ashish Ghoda

All rights reserved. No part of the contents of this book may be reproduced or transmitted in any form or by any means without the written permission of the publisher.

ISBN: 978-0-7356-5896-7

1 2 3 4 5 6 7 8 9 LSI 6 5 4 3 2 1

Printed and bound in the United States of America.

Microsoft Press books are available through booksellers and distributors worldwide. If you need support related to this book, email Microsoft Press Book Support at [mspinput@microsoft.com](mailto:mspinput@microsoft.com). Please tell us what you think of this book at <http://www.microsoft.com/learning/booksurvey>.

Microsoft and the trademarks listed at <http://www.microsoft.com/about/legal/en/us/IntellectualProperty/Trademarks/EN-US.aspx> are trademarks of the Microsoft group of companies. All other marks are property of their respective owners.

The example companies, organizations, products, domain names, email addresses, logos, people, places, and events depicted herein are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

This book expresses the author's views and opinions. The information contained in this book is provided without any express, statutory, or implied warranties. Neither the authors, Microsoft Corporation, nor its resellers, or distributors will be held liable for any damages caused or alleged to be caused either directly or indirectly by this book.

**Acquisitions and Developmental Editor:** Russell Jones

**Production Editor:** Kristen Borg

**Editorial Production:** S4Carlisle Publishing Services

**Technical Reviewer:** Vikas Sahn

**Copyeditor:** Becka McKay

**Indexer:** Denise Getz

**Cover Design:** Twist Creative • Seattle

**Cover Composition:** Karen Montgomery

**Illustrator:** S4Carlisle Publishing Services

*To Nimish and to my mother, for being my inspiration and strength.*

—MAMTA DALAL

*I dedicate this book to my grandparents (Nayansukhray and Kumud Ghoda, Mahavir and Sarla Majmudar), parents (Jitendra and Varsha Ghoda), sister (Kruti Vaishnav), and lovely family (Pratixa, Gyan, and Anand Ghoda) whose blessings, sacrifice, continuous support, and encouragement enabled me to achieve the dream.*

—ASHISH GHODA



# Contents at a Glance

---

	<i>Introduction</i>	<i>xiii</i>
<b>PART I</b>	<b>XAML BASICS</b>	
CHAPTER 1	Introducing XAML	3
CHAPTER 2	Object Elements and Attributes	19
CHAPTER 3	XAML Properties and Events	49
<b>PART II</b>	<b>ENHANCING USER EXPERIENCE</b>	
CHAPTER 4	Markup Extensions and Other Features	87
CHAPTER 5	Resources, Styles, and Triggers	101
<b>PART III</b>	<b>XAML USER INTERFACE CONTROLS</b>	
CHAPTER 6	Layout and Positioning System	129
CHAPTER 7	Form and Functional Controls	171
<b>PART IV</b>	<b>CONTENT INTEGRATION AND ANIMATION</b>	
CHAPTER 8	Data Binding	213
CHAPTER 9	Media, Graphics, and Animation	245
<b>PART V</b>	<b>APPENDIXES</b>	
APPENDIX A	Major Namespaces and Classes	289
APPENDIX B	XAML Editors and Tools	299
	<i>Index</i>	<i>303</i>



# Contents

*Introduction* ..... *xiii*

**PART I      XAML BASICS**

---

**Chapter 1    Introducing XAML** ..... **3**

    Windows Presentation Foundation (WPF) ..... 4

    XAML—A Declarative Language for .NET Applications ..... 4

        XAML Is Part of the Microsoft Open Specification Program (OSP) . . 6

        XAML Structure ..... 6

        Dynamic User Interface. .... 7

        Decouple Control Style Definitions ..... 8

        Customized Design of XAML Controls ..... 9

        Integration with Code-Behind to Control Behavior. .... 9

        Inline Code ..... 12

    Silverlight ..... 13

    The Microsoft .NET Framework ..... 14

        Design-Time Components ..... 15

        Runtime Cross-Platform Components ..... 16

    Summary. .... 17

---

**What do you think of this book? We want to hear from you!**  
Microsoft is interested in hearing your feedback so we can continually improve our books and learning resources for you. To participate in a brief online survey, please visit:  
[microsoft.com/learning/booksurvey](https://microsoft.com/learning/booksurvey)

## PART II     ENHANCING USER EXPERIENCE

viii Contents

Type Converters versus Markup Extensions. . . . .	98
XAML Services. . . . .	99
Security in XAML. . . . .	99
Summary. . . . .	100

## **Chapter 5 Resources, Styles, and Triggers 101**

Resources . . . . .	101
Types of Resources . . . . .	102
Static Resources . . . . .	102
Defining Static Resources Using XAML . . . . .	102
Defining Static Resources Programmatically. . . . .	104
Dynamic Resources . . . . .	105
When to Use Which Resource . . . . .	106
How Static and Dynamic Resources Work. . . . .	106
Defining <i>ResourceDictionary</i> Files . . . . .	107
Merged Resource Dictionaries . . . . .	108
Scope and Hierarchy of Resources. . . . .	109
Styles. . . . .	111
Defining Styles . . . . .	112
Implicit Styles . . . . .	115
Inheriting Styles . . . . .	116
The Silverlight Toolkit Styles. . . . .	117
Styles vs. Control Templates. . . . .	117
More on Styles . . . . .	117
The generic.xaml File. . . . .	119
Triggers. . . . .	120
Troubleshooting Resources, Styles, and Triggers . . . . .	126
Summary. . . . .	126

## PART III XAML USER INTERFACE CONTROLS

---

<b>Chapter 6</b>	<b>Layout and Positioning System</b>	<b>129</b>
	The Layout System .....	130
	XAML Layout and Positioning Controls .....	135
	Common Sizing and Positioning Properties .....	160
	Summary .....	170
<b>Chapter 7</b>	<b>Form and Functional Controls</b>	<b>171</b>
	Action Controls .....	172
	The <i>ButtonBase</i> Class .....	172
	Text Editing Controls .....	182
	The <i>TextBoxBase</i> Class .....	182
	Functional Controls to Improve Usability .....	194
	Functional Controls to Control and Monitor Behavior .....	205
	The <i>RangeBase</i> Class .....	205
	Summary .....	210

## PART IV CONTENT INTEGRATION AND ANIMATION

---

<b>Chapter 8</b>	<b>Data Binding</b>	<b>213</b>
	Data Sources .....	213
	Data Binding .....	215
	Setting the Binding Source .....	216
	MultiBinding .....	221
	Binding to Data from a Database .....	221
	Binding Modes .....	227
	Example of Two-Way Binding with <i>TextBox</i> .....	227
	Source Updates .....	228
	Data Templating, Conversion, and Validation .....	228
	Data Templating .....	230
	Data Conversion .....	231
	Data Validation .....	233



## PART V    APPENDIXES

---

<b>Appendix A   Major Namespaces and Classes</b>	<b>289</b>
Commonly Used Namespaces and Classes in WPF . . . . .	289
Commonly Used Namespaces and Classes in Silverlight . . . . .	293
 <b>Appendix B   XAML Editors and Tools</b>	 <b>299</b>
Editors . . . . .	299
Kaxaml . . . . .	299
XAML Cruncher . . . . .	299
XamlPad . . . . .	300
XamlPadX . . . . .	300
Tools . . . . .	300
 <i>Index</i>	 303

---

**What do you think of this book? We want to hear from you!**

Microsoft is interested in hearing your feedback so we can continually improve our books and learning resources for you. To participate in a brief online survey, please visit:

[microsoft.com/learning/booksurvey](https://microsoft.com/learning/booksurvey)

# Introduction

XAML is ubiquitous today. Whether with Silverlight, WPF, WF, various XPS formats, or XML-based formats, XAML is being used in a whole lot of Microsoft platform-based technologies. Though based on XML, XAML is unlike most other markup languages, because it is strongly linked to CLR assemblies through its objects.

Microsoft originally intended XAML to be a new and much more malleable and adaptable user interface (UI) description language for the .NET Framework through a technology named Windows Presentation Foundation (formerly called WinFX). From that specific beginning, XAML has not only outgrown that original goal, but achieved far more.

Recently, WPF has begun to supersede Windows Forms as the preferred development target. XAML's support for rich web interfaces, media streaming, and data-driven Line-of-Business (LOB) applications has made Silverlight a popular application platform in the web development community. The upcoming version of the Windows operating system, Windows 8, also includes extensive support for XAML.

This book introduces you to XAML and explains its syntax and constructs. It then explores various concepts, including XAML elements, properties, data binding, and so forth. Although the book does not provide exhaustive coverage of every XAML feature, it does offer essential guidance in using the key XAML functionality; you'll gain a strong foundation for designing rich and powerful user interfaces and applications using either WPF or Silverlight.

Beyond the explanatory content, each chapter includes procedural examples and downloadable sample projects that you can explore and expand for your own projects.

## Who Should Read This Book

---

This book is aimed at proficient developers using the .NET platform, who understand the core concepts of XAML. It is especially useful for programmers looking to work with new or existing WPF or Silverlight applications. Although most readers will have some experience with XAML, the book is also suitable for those who are new to XAML but wish to learn XAML development.

## Assumptions

This book expects that you have at least a minimal understanding of .NET-based WPF and Silverlight development with C# or Visual Basic. The book also assumes that you have a basic knowledge of SQL Server and XML.

If you have not yet gained familiarity with Silverlight or WPF, you might consider reading the following books:

- Ashish Ghoda's *Introducing Silverlight 4* (Apress, 2010), or Laurence Moroney's *Microsoft Silverlight 4 Step by Step* (Microsoft Press, 2010)
- Adam Nathan's *WPF 4 Unleashed* (Sams, 2010)

## Who Should Not Read This Book

---

If you are completely unfamiliar with WPF and Silverlight, or if you're not comfortable reading and writing C# or Visual Basic code, this book is not for you. This book does not include a detailed explanation of the Model-View-ViewModel (MVVM) pattern; if you're looking for that information, take a look at:

- Raffaele Garofalo's *Building Enterprise Applications with Windows Presentation Foundation and the Model View ViewModel Pattern* (Microsoft Press, 2011)
- Gary Hall's *Pro WPF and Silverlight MVVM: Effective Application Development with Model-View-ViewModel (Expert's Voice in WPF)* (Apress, 2010)

This book also does not cover XAML for Windows 8—the timing of this edition of book precluded including that information with any reasonable hope of accuracy. However, based on our current level of information, the majority of the basic XAML concepts should remain the same for the future Windows 8 platform.

## Organization of This Book

---

This book is divided into four sections, each of which focuses on a different aspect or set of features within XAML.

- Part I, “XAML Basics,” introduces the .NET Framework and provides a quick overview of XAML fundamental concepts and classes, including object elements, attributes, properties, and events.

- Part II, “Enhancing User Experience,” describes the various language features such as markup extensions, resources, and styles.
- Part III, “XAML User Interface Controls,” describes the layout system and various XAML controls.
- Part IV, “Content Integration and Animation,” delves into data binding, media, graphics, and animation.

## Finding Your Best Starting Point in This Book

The different sections of the *XAML Developer Reference* cover a wide range of technologies associated with the Microsoft .NET Framework library and design and development tools. Depending on your needs and your existing understanding of the Microsoft .NET Framework, WPF, Silverlight, data binding, and design and development tools, you may wish to focus on specific areas of the book.

If you are	Follow these steps
New to XAML development	Focus on Parts I and III, or read through the entire book in chapter sequence. To get an overview of different XAML controls used in various samples throughout this book, read Chapters 6 and 7, which introduce layout and form and functional XAML controls.
Familiar with XAML	Briefly skim Parts I and III if you need a refresher on the core concepts. To get an overview of different XAML controls used in various samples throughout this book, read Chapters 6 and 7, which introduce layout and form and functional XAML controls. Read up on markup extensions, styles, and other features in Parts II and IV.

Most of the book’s chapters include hands-on samples that let you try out the concepts covered in that chapter. No matter which chapters or parts you choose to focus on, be sure to download and install the sample applications on your system.

## Conventions and Features in This Book

This book presents information using conventions designed to make the information readable and easy to follow.

- In most cases, the book includes examples that are XAML markup–based. Although you will see some minimal C# code to show the connection of the XAML to the code-behind code, the exercises rarely delve deeply into any code-behind.

- Boxed elements with labels such as “Note” provide additional information or alternative methods for completing a step successfully.
- Text that you need to type (apart from code blocks) appears in **bold**.
- A plus sign (+) between two key names means that you must press those keys at the same time. For example, “Alt+Tab” means that you hold down the Alt key while you press the Tab key.
- A vertical bar between two or more menu items (such as File | Close), means that you should select the first menu or menu item, then the next, and so on.

## System Requirements

---

You will need the following hardware and software to complete the practice exercises in this book:

- One of the following: Windows Vista with Service Pack 2 (except Starter edition), Windows XP with Service Pack 3 (except Starter edition), or Windows 7.
- Microsoft .NET Framework 4.0 or 3.5 SP1 (4.0 is recommended)
- Silverlight 4 SDK, toolkit, and run time (including developer run time)
- SQL Server 2008 Express edition or higher (2008 or R2 release), with SQL Server Management Studio 2008 Express or higher (included with Visual Studio; Express editions require separate download)
- Visual Studio 2010, any edition (multiple downloads may be required if using Express edition products)
- Microsoft Expression Blend 4
- Computer that has a 1.6 GHz or faster processor (2 GHz or above recommended)
- Minimum 1 GB (32-bit) or 2 GB (64-bit) RAM (Add 512 MB if running in a virtual machine or SQL Server Express editions, more for advanced SQL Server editions)
- 3.5 GB of available hard disk space
- 5400 RPM hard disk drive
- DirectX 9–capable video card running at 1024 x 768 or higher-resolution display

- DVD-ROM drive (if installing Visual Studio and Expression Blend from DVD)
- Internet connection to download software or chapter examples

Depending on your Windows configuration, you might require Local Administrator rights to install or configure Visual Studio 2010 and SQL Server 2008 products.

## Code Samples

---

Most of the chapters in this book include projects or code snippets that let you interactively try out the new material discussed in the main text. You can download all the sample code from this link:

<http://www.microsoftpressstore.com/title/9780735658967>

Follow the instructions to download the XAML\_Developer\_Reference\_samples.zip file.



**Note** In addition to the code samples, your system should have Visual Studio 2010 and SQL Server 2008 installed. The following instructions use SQL Server Management Studio 2008 to set up the sample database used with the practice examples. If available, install the latest service packs for each product.

## Installing the Code Samples

Follow these steps to install the code samples on your computer so that you can use them with the exercises in this book:

1. Unzip the XAML\_Developer\_Reference\_samples.zip file that you downloaded from the book's website. (Name a specific directory along with directions to create it, if necessary.)
2. If prompted, review the displayed end user license agreement. If you accept the terms, select the Accept option, and then click Next.



**Note** If the license agreement doesn't appear, you can access it from the same webpage from which you downloaded the XAML\_Developer\_Reference\_samples.zip file.

3. Attach the Northwind sample database to your instance of SQL Server 2008.

## Using the Code Samples

The folder created by the Setup.exe program contains three subfolders:

- **Chapters** Example projects referenced in each chapter appear in this folder. Each chapter appears as subfolder with chapter number. Each chapter folder may include one or more sample projects related to that chapter. The chapter may contain separate projects for WPF and Silverlight. Follow the instructions given in the chapter to run the project. Some of the chapters may include one or more XAML file for individual samples. Some of these projects are incomplete, and will not run without following the steps indicated in the associated chapter.
- **Snippets** Fragmented or partial code snippets that are included in the chapter are included in text files. These can be copied and pasted into existing projects or applications and then executed.
- **Sample Database** This folder contains the SQL script used to build the sample database. The instructions for creating this database appear earlier in this Introduction.

To access the example project of a particular chapter, browse to the appropriate chapter folder in the Chapters folder, and open the project file.

## Acknowledgments

---

### Mamta Dalal:

This book is the culmination of the efforts of a number of people. Therefore, I'd like to thank the editorial and copyedit team of Microsoft and O'Reilly—in particular, our editor Russell Jones, without whom this book would not have been possible. I am also grateful to Kristen Borg, our production editor at O'Reilly; and Diane Kohnen and her amazing copyediting team. I would also like to thank my coauthor Ashish Ghoda for his valuable collaboration and strong support. Vikas Sahni, our technical reviewer, also deserves a strong vote of thanks for his feedback, which went a long way toward making this book better. I thank my parents for having believed in me and for encouraging me to nurture my skills.

I would also like to take this opportunity to thank the awesome .NET, WPF, and Silverlight communities at the MSDN forums and at Stackoverflow.com. The latter in particular has been a tremendous source of enlightenment for me. Thank you to Jeff Atwood and Joel Spolsky for having created this wonderful site.

Finally, I thank my husband, Nimish, for his constant encouragement, understanding, love, and support.

**Ashish Ghoda:**

Working with the Microsoft Press and O'Reilly teams and my coauthor for this book was a great experience. The support, positive attitude, and constructive feedback from the Microsoft Press editorial and production teams and from our technical reviewer—Vikas Sahni—made this project run smoothly.

My special thanks goes to Russell Jones—senior editor of Microsoft Press division—for giving me the opportunity to help write this book and for remaining confident that we could finish the book in the given time frame, despite some unexpected personal challenges faced by both Mamta and myself.

It's challenging when the authors of a work are located in different countries. Mamta Dalal, coauthor of this book, deserves full credit for her cooperation and efforts to keep the content in sync while working remotely.

With blessings from God and encouragement from my grandparents, parents, and in-laws, I was able to accomplish this task successfully. My wife, Pratixa, and two God-gifted sons, Gyan and Anand, have continued their support so that I could finish a fourth consecutive book. I thank my family for their cooperation and encouragement and for their faith in me during this difficult endeavor.

## Errata & Book Support

---

We've made every effort to ensure the accuracy of this book and its companion content. Any errors that have been reported since this book was published are listed on our Microsoft Press site:

*<http://www.microsoftpressstore.com/title/9780735658967>*

If you find an error that is not already listed, you can report it to us through the same page.

If you need additional support, email Microsoft Press Book Support at *[mspinput@microsoft.com](mailto:mspinput@microsoft.com)*.

Please note that product support for Microsoft software is not offered through the addresses above.

## We Want to Hear from You

---

At Microsoft Press, your satisfaction is our top priority, and your feedback our most valuable asset. Please tell us what you think of this book at:

*<http://www.microsoft.com/learning/booksurvey>*

The survey is short, and we read every one of your comments and ideas. Thanks in advance for your input!

## Stay in Touch

---

Let's keep the conversation going! We're on Twitter: *<http://twitter.com/MicrosoftPress>*

**PART 1**

# XAML Basics

<b>CHAPTER 1</b>	Introducing XAML . . . . .	<b>3</b>
<b>CHAPTER 2</b>	Object Elements and Attributes . . . . .	<b>19</b>
<b>CHAPTER 3</b>	XAML Properties and Events . . . . .	<b>49</b>



# Introducing XAML

## In this chapter:

- Windows Presentation Foundation (WPF)
- XAML—A Declarative Language for .NET Applications
- Silverlight
- The Microsoft .NET Framework
- Summary

Object-oriented and service-oriented programming models (along with language- and environment-independent features) lie at the core of the .NET Framework architecture. Since the release of .NET Framework 3.0, Microsoft has added several important components to support the unified programming and deployment model:

- **A presentation layer** Windows Presentation Foundation (WPF)
- **A messaging and communication services layer** Windows Communication Foundation (WCF)
- **Workflow management** Windows Workflow Foundation (WF)

You use the Windows Presentation Foundation (WPF) framework libraries along with the XML-based *eXtensible Application Markup Language (XAML)* declarative markup language to define and develop next-generation, abstracted, dynamic, rich, and interactive user interface layers that provide data-integration capabilities and comprehensive support for multimedia, graphics, animation, and documents.

The **eXtensible Application Markup Language (XAML, pronounced *zammel*)**—a declarative XML-based markup language—is at the center of the declarative user interface (UI) WPF framework. It is a language for describing an *abstracted*—externalized and decoupled—user interface layer. The current .NET Framework has extended XAML as its core user interface definition language to define user interfaces not only for WPF and Silverlight applications, but also for the custom activity libraries of WF 4.0–based workflows.

# Windows Presentation Foundation (WPF)

---

WPF supports the development of rich and interactive Windows desktop applications that can provide sophisticated and realistic user experiences. WPF is built upon a very different architecture than Windows Forms. The key architectural differences are:

- WPF introduces a new user interface XML-based declarative markup language—XAML—that can support layout, styles, resources, and control templates to simplify and standardize management of the visual appearance of the user interface. XAML also supports properties and events that developers can handle in code-behind code to control its behavior.
- WPF provides a new presentation framework that integrates XAML for user interface design. The framework supports a unified programming model that includes data binding capabilities to develop data-driven applications as well as media integration, 2-D and 3-D vector graphics, document integration, text, and typography.
- WPF provides a set of .NET Framework libraries for the presentation core that are mainly derived from the *System.Windows* namespace. These libraries handle integration of the XAML-based user interface with the managed code-behind, including enhanced properties and events integration, such as *dependency properties* and *routed events* (topics you'll explore later in this book).
- The new Media Integration Layer (MIL) provides a rendering engine for WPF applications built upon DirectX. The tight integration with DirectX means that WPF has high-performance rendering of the visual interface that can take advantage of hardware acceleration using the graphics processing units (GPUs) that most modern computers have, which reduces the load on the central processing unit (CPU). This is a very different approach than that taken in Windows Forms applications, where the .NET Framework uses the User32 DLL to render standard Windows Forms user interface elements and uses older Graphics Device Interface (GDI) to render graphics. Figure 1-1 illustrates the differences between the visual interface rendering approaches for WPF and Windows Forms applications.



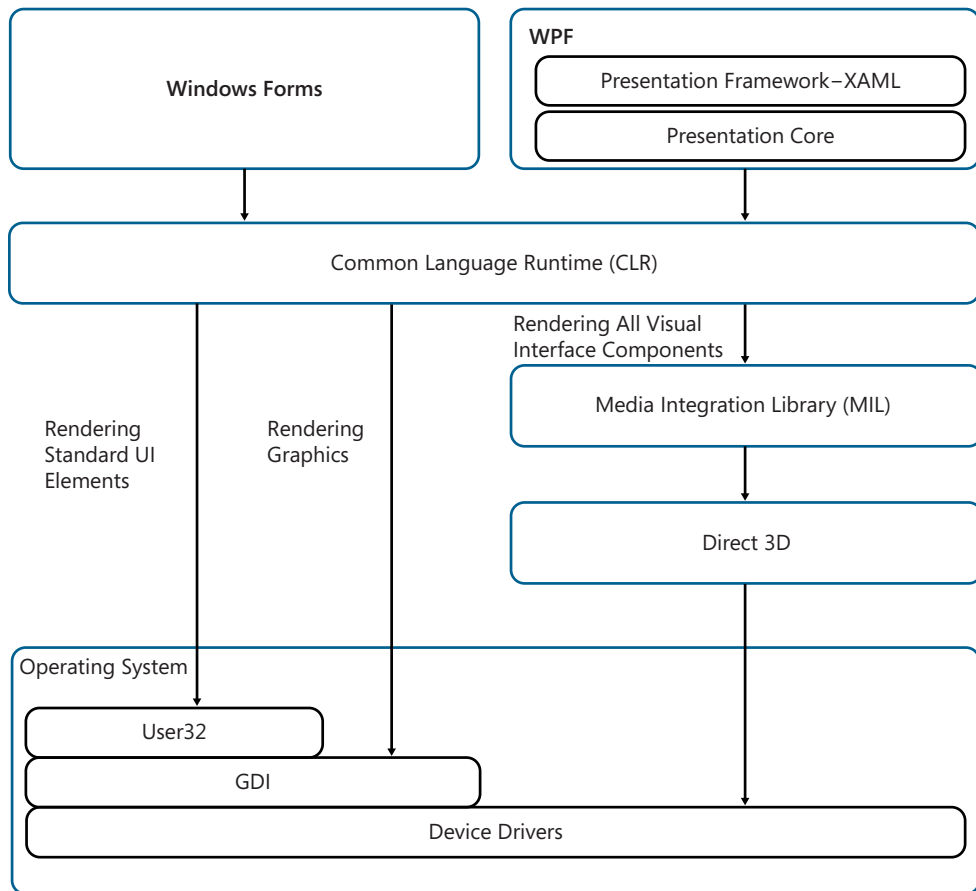
**More Info** Visit MSDN at <http://msdn.microsoft.com/en-us/library/ms750441.aspx> to get more details on the WPF architecture.

## XAML—A Declarative Language for .NET Applications

---

As mentioned earlier, XAML is at the center of the declarative user interface (UI) WPF framework because it implements the abstracted user interface layer. XAML is becoming a core UI definition language for .NET Framework-based applications. Using XAML, you can define and develop user interfaces for WPF and Silverlight applications and custom workflow activities for WF version 4.0.

You define and implement these user interfaces using a set of XAML controls provided as part of the WPF framework. These XAML controls are derived from a set of WPF presentation framework classes that can be hosted in either a window (WPF applications) or a page (Silverlight applications) to render the defined user interface at runtime using a XAML parser.



**FIGURE 1-1** Rendering WPF and Windows Forms applications.



**Caution** Not all XAML controls are interoperable between WPF, Silverlight, and WF applications. In addition, the XAML parsers for each platform are also different. You will need to use and set appropriate WPF, Silverlight, and WF platform specific-XAML controls and compile applications using the corresponding platform.

## XAML Is Part of the Microsoft Open Specification Program (OSP)

You are probably aware that XAML is currently a Microsoft domain-specific language. To provide transparency and simplify the development of XAML applications by the broader developer community, Microsoft published the technical specification of XAML Object Mapping for WPF and Silverlight in March 2008, under its Open Specification Promise (OSP) program. Microsoft is committed to maintaining those specifications.

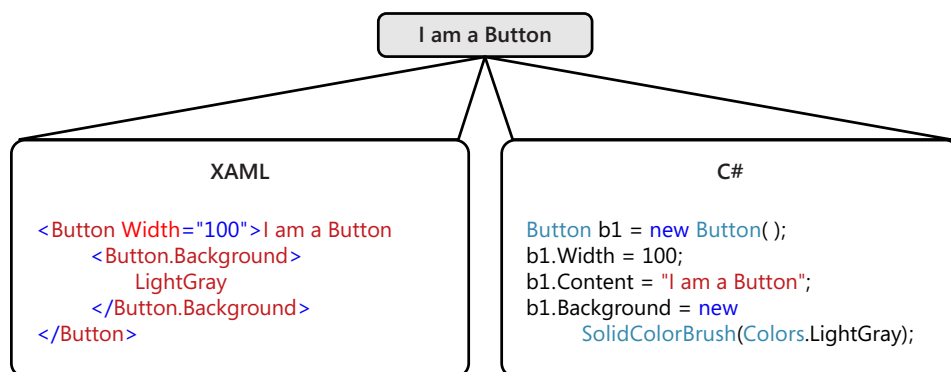
The XAML technical specification documentation provides details on XAML's data model for types, object hierarchies, and the techniques for mapping between XML and the object hierarchy data model. It also documents the WPF and Silverlight vocabulary of types that can be used with XAML specifications. Developers can use the WPF and Silverlight XAML technical specification documentation in conjunction with publicly available standard specifications, computer language design, and implementation art to fully understand and take advantage of XAML.



**More Info** Visit MSDN at <http://msdn.microsoft.com/en-us/library/dd361847.aspx> to download the various releases of XAML Object Mapping, and the WPF and Silverlight technical specification documentation.

## XAML Structure

Figure 1-2 provides a quick overview of defining a button. In the example, the button width is set to *100*, the button background color is set to *LightGray*, and the content (the button label) is set to "I am a Button" in XAML. The example also shows an identical *Button* object created in C#, with its related properties set in code.



**FIGURE 1-2** Defining a *Button* object and its properties in XAML and in C# code-behind.

A XAML file has a *.xaml* file extension. As shown in Figure 1-2, any XAML file consists of XML-like structured information that defines the relationships among various XAML controls. At runtime, these controls render as an object tree to create the user interface. In other words, XAML itself is an abstraction—it simply describes objects. This abstraction lets XAML serve as the UI description

language for several different .NET application types (WPF, Silverlight, and WF). The properties you define within the XAML elements (such as the *Width*, *Content*, and *Background* properties of the *Button* control in Figure 1-2) control the look and feel of the particular user interface object represented by that XAML element. You can also determine how or whether a control binds with data. When you bind a control, it can display information (often from a database) unavailable at design time, and obtained only at runtime.

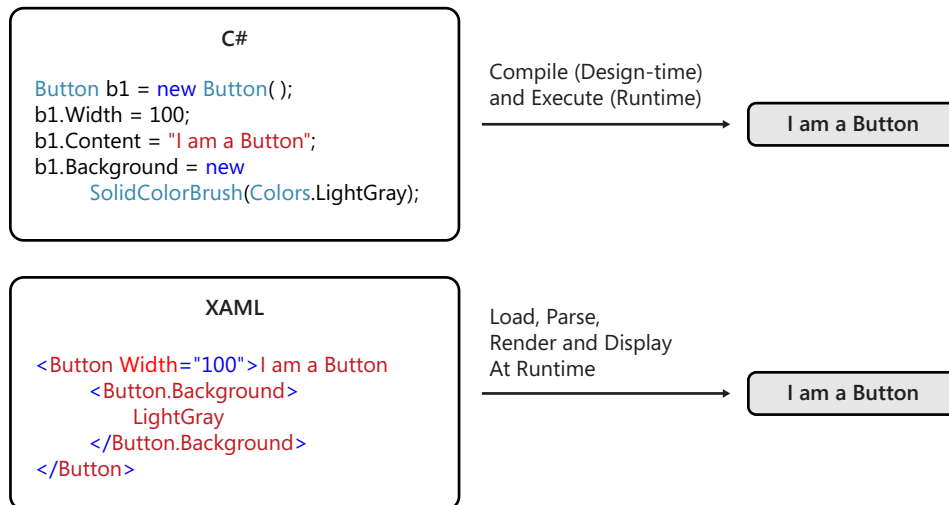


**More Info** See Chapter 2, “Object Elements and Attributes,” for more details on XAML syntax, XAML object elements, and attributes.

## Dynamic User Interface

As shown in Figure 1-3, the key difference and advantage of using XAML—compared to building the user interface by creating and adding the controls in code-behind—is that XAML provides a declarative and separately compiled and rendered way of describing the user interface. User interface controls defined in code are described at design time and executed at runtime. In contrast, controls defined in XAML are stored separately from compiled code in .xaml files. At runtime, the XAML file is loaded and parsed by a XAML parser, and the user interface is then rendered dynamically. Thus if you change the user interface within a XAML file and redeploy it, the updated XAML content will be parsed and rendered; any changes in the user interface definition will be reflected in the user interface.

XAML’s capability to develop an externalized and loosely coupled user interface enables developers to develop and modify the user interface without affecting the underlying program code and without recompiling the project for each UI change, which can significantly reduce the overall effort required for application development and testing.



**FIGURE 1-3** Defining a *Button* object with its properties in XAML and using C# in code-behind.

When working with XAML, remember that the WPF XAML parser is full-featured, whereas the Silverlight XAML parser ships with a more limited feature set. As mentioned earlier, not all XAML controls are interoperable between WPF, Silverlight, and WF applications. You will need to use the appropriate WPF, Silverlight, and WF platform-specific XAML controls and compile applications using the specific platform to which you want to deliver.




**More Info** Visit MSDN at <http://msdn.microsoft.com/en-us/library/cc917841.aspx> for more details on the differences between WPF and the Silverlight XAML parser.

## Decouple Control Style Definitions

Applications should maintain consistency throughout to give users a predictable experience, including using the same color set, fonts, font sizes, and styles. Typically, ensuring this consistency can become quite challenging when you are using multiple controls of similar types in single or multiple XAML files within the same application, or across multiple applications. However, the WPF and Silverlight platforms help, because it provides the capability to easily externalize and decouple *style sheets*, which XAML elements can then reference from within XAML files to help maintain a consistent user experience. The approach and capability is similar to the Cascading Style Sheets (CSS) approach used in standard HTML web applications.

Figure 1-4 demonstrates how you can define a style within a XAML file as a resource or as an external resource file, and apply that style to a *Button* control.

 **Defining a Style**

```
<Style x:Key="ButtonStyle5" TargetType="Button">
  <Setter Property="Foreground" Value="Black"/>
  <Setter Property="Background" Value="Green"/>
  <Setter Property="FontStyle" Value="Italic"/>
  <Setter Property="FontFamily" Value="Verdana"/>
  <Setter Property="FontSize" Value="16"/>
</Style>
```

**Applying a Style**

```
<Button Style="{StaticResource ButtonStyle5}"
  Width="115" Content="Button"/>
```

**FIGURE 1-4** Defining and applying styles to XAML controls.



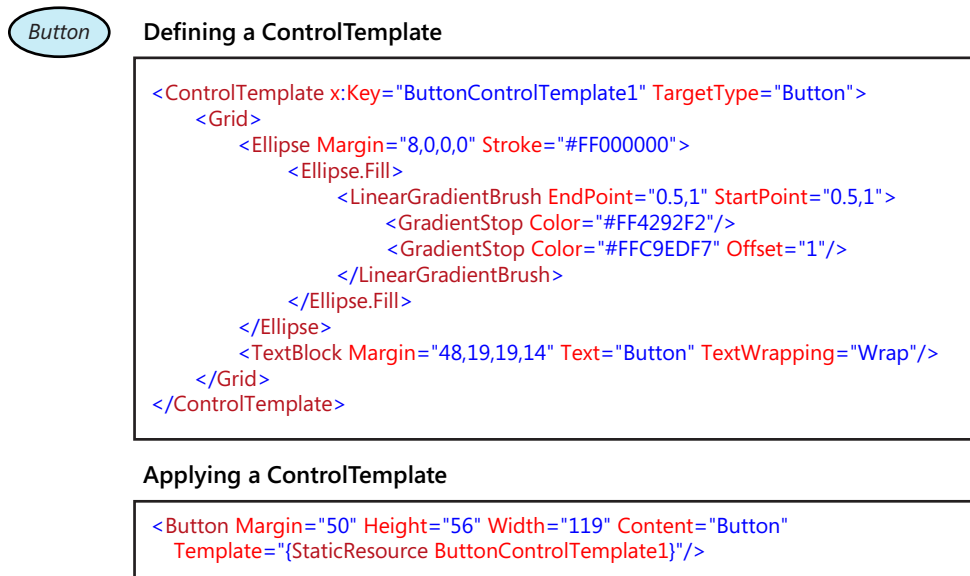
**More Info** See Chapter 5, "Resources, Styles, and Triggers," for more details on styles and resources for XAML.

## Customized Design of XAML Controls

One of the biggest advantages of WPF's separation of the visual appearance of controls defined in XAML from business logic implemented mainly in code is that a designer not only controls the common styles of controls but can also alter the default look and feel of the control. For example, you might change the default look and feel of a button to make it look like a star! In WPF you can use a *ControlTemplate* to define the visual structure and behavior of a control without affecting its functionality.

Each control can exist in a number of possible states, such as disabled, having the input focus, a state where the mouse is hovering over it, and so on. A control template lets you define what a control looks like in each of these states. Sometimes this is referred to as changing the look and feel of the control, because changing the visual appearance of each state alters how a user sees and experiences a control.

Figure 1-5 demonstrates how you can define a control template to change the appearance of a *Button* control to make it look like an ellipse.



**FIGURE 1-5** Defining and applying a control template to XAML controls.

## Integration with Code-Behind to Control Behavior

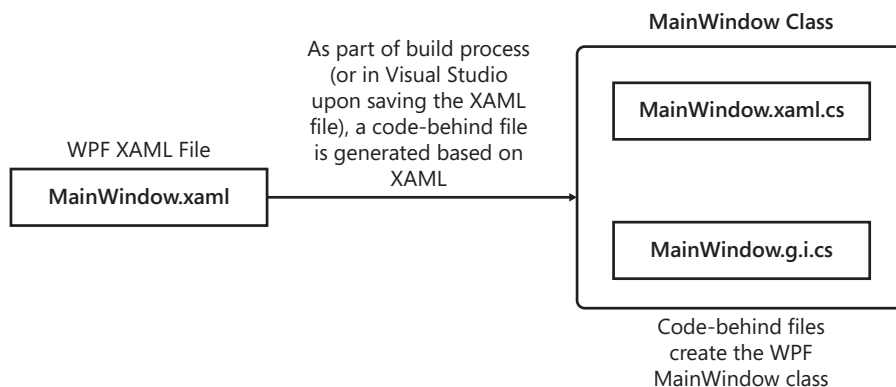
In general, markup languages such as HTML are mainly limited to defining the look and feel of the user interface; most markup languages cannot define the *behavior* of the user interface by controlling user interactions and defining various application actions. The typical way to implement some level of business logic within an HTML file is to use a scripting language such as JavaScript or VBScript.

In contrast to HTML, XAML was specifically developed for use with .NET Framework components. It can use the .NET Framework platform and the Microsoft design and development tools and extend its capabilities because it's not limited simply to defining the user interface. It also enables interaction by integrating XAML controls with managed code such as C# and VB .NET, and even dynamic languages such as Ruby and Python.



**More Info** See the article “Creating Interactive Bing Maps with Silverlight and IronRuby,” at <http://msdn.microsoft.com/en-us/magazine/ee291739.aspx> for an example of how IronRuby dynamic language integrates events of XAML objects to implement required business logic.

Each XAML file for WPF, Silverlight, or WF project has a corresponding *code-behind* file, which Microsoft development tools such as Visual Studio or Expression Blend create for you automatically. However, a third file type is associated with the XAML file. Figure 1-6 illustrates the full class implementation for the MainWindow XAML file of a standard WPF project created using either Visual Studio or Expression Blend.



**FIGURE 1-6** Full class implementation of XAML.



**Note** As defined on MSDN, “code-behind is a term used to describe the code that is joined with markup-defined objects, when a XAML page is markup-compiled.” See <http://msdn.microsoft.com/en-us/library/aa970568.aspx> to get more information on the code-behind capabilities of XAML.

If you create a WPF application project by selecting WPF Application template in Visual Studio, you will get a default MainWindow.xaml file. If you expand the MainWindow.xaml file in the Visual Studio Solution Explorer, you will see an associated code-behind file named either MainWindow.xaml.cs file (when you create a C# WPF project) or MainWindow.xaml.vb (when you create a Visual Basic WPF project). This code-behind class is usually used to manage events and as a gateway to integrate with other application components and services to implement the business logic.

Now, open this code-behind file in the code editor. Locate the class constructor and right-click the *InitializeComponent()* method. Select the Go To Definition option from the shortcut menu. You will see that the *InitializeComponent* definition code opens a *MainWindow.g.i.cs* file. *MainWindow.g.i.cs* is a generated file based on the XAML defined in the *MainWindow.xaml* file. Any objects in the XAML file that have an *x:Name* cause the creation of a class member in the generated file.

The following code snippet demonstrates the default *MainWindow.xaml.cs* file of the WPF application and the *InitializeComponent()* method (in bold font) within the class constructor:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
namespace WpfApplication1
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }
    }
}
```

XAML defines the language features *x:Class*, *x:Subclass*, and *x:ClassModifier* directives, which (as you will explore more deeply in Chapter 2) enable integration of the XAML markup file with the code-behind partial class. You must derive the partial class defined in the root element of the XAML markup file using the *x:Class* attribute. This class usually gets defined automatically by Visual Studio, using the naming convention *<XAMLFileName.xaml>.cs* or *<XAMLFileName.xaml>.vb*, depending on which .NET language you're using. The following code snippet shows the definition of the *x:Class* attribute (in bold font) defined in the *Window* root element of the *MainWindow.xaml* file of the WPF application.

```
<Window x:Class="WpfApplication1.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="MainWindow" Height="350" Width="525">
    <Grid>
    </Grid>
</Window>
```



**Note** Because XAML is a declarative language, it can contain data binding, state management, triggers, and so on as part of the UI definition. That means that design patterns such as the Model-View-Controller (MVC) and Model-View-Presenter (MVP) that were developed for service-oriented applications are not the best-fitting patterns for WPF-based applications. Instead, a new design pattern called the Model-View-View-Model (MVVM) pattern has been developed to define the user interface layer for XAML-based applications. Although MVVM was largely derived from the concept of MVC and MVP patterns, it differs by defining a view model that represents both a data model and behavior for views, and allows views to bind to the view model declaratively within XAML. Visit <http://msdn.microsoft.com/en-us/magazine/dd419663.aspx> to get an overview of how you can develop WPF applications using the MVVM design pattern. Also see <http://weblogs.asp.net/dwahlin/archive/2009/12/08/getting-started-with-the-mvvm-pattern-in-silverlight-applications.aspx> to get an overview on how to develop Silverlight applications using MVVM.

## Inline Code

The WPF XAML namespace also supports an additional `x:Code` directive element that can contain inline programming code (in C# or Visual Basic) to implement business logic directly within the XAML file. Programming code within the `x:Code` element must be entered inside a `<CDATA[...]>` segment so that it will be processed as code rather than as XML by the XAML parser.

The following example implements the `Click` event of a `Button` control within the XAML file as inline code (in bold font). The code is written in C#, and is defined right next to the definition of the `Button` control, but within an `x:Code` element:

```
<Button Name="button1" Click="button1_click">Click Me!</Button>
<x:Code>
  <![CDATA[
    void button1_click(object sender, RoutedEventArgs e)
    {
      button1.Content = "Inline Code Works!!";
    }
  ]]>
</x:Code>
```



**Warning** Despite the existence of the `<x:code>` element, inline coding within XAML is not considered a best practice, and its use is not recommended except in special circumstances. It's definitely not the best way to implement complex business logic. Inline code has some limitations that make implementing reusable code across a project considerably more challenging. In addition, it's more difficult to code, maintain, and support complex business logic in inline code.

The inline code must be defined within the XAML file. The scope of inline code is limited to the scope of the partial class created for that particular XAML instance.

You cannot use *using* (C#) or *Imports* (VB.NET) statements. Instead, you must fully qualify references to code entities outside the partial class.

The `<x:Code>` element must be an immediate child element of the root element of the XAML production. Moreover, although XAML itself has the advantage of abstracting the user interface definition of the application from the implementation of the business logic, inline code does not provide that abstraction, because it's defined directly within the XAML file.



**Caution** The `x:Code` directive (and thus inline coding) is supported only by the WPF XAML parser—it is not supported by the Silverlight XAML parser. Therefore, you cannot implement inline coding in Silverlight applications. The Silverlight XAML parser also does not guarantee preservation of `CDATA` segment content.

## Silverlight

---

Silverlight is an extension of the .NET Framework–based technology platform to develop cross-browser, cross-platform, and cross-device Rich Internet Applications (RIAs). RIAs are web applications that have features and functionality similar to traditional desktop applications, including rich and interactive user interfaces.

You can deploy Silverlight applications as plug-ins (in both in-browser and out-of-browser modes) that run in a sandboxed environment. Silverlight is built upon lightweight components of the .NET Framework that are a subset of the full WPF libraries. Silverlight applications do not require users to perform a full install of the .NET Framework; instead, users need to install only a small Silverlight plug-in on their Windows or Mac (Intel processor–based) computers, or Windows Phone 7 mobile devices.



**Note** To install the latest version of Silverlight and get the latest information on Silverlight, visit Microsoft's official Silverlight website at <http://www.silverlight.net/getstarted/>.

Like WPF Windows applications, the declarative XAML markup language used by Silverlight is at the center of the declarative user interface (UI) framework. You can use the same Microsoft development tools (Visual Studio and Expression Blend) to define Silverlight user interfaces in XAML and you can implement business logic using standard .NET languages. However, there is a significant difference between the set of XAML controls available for WPF and those available for Silverlight. In addition, the Silverlight platform has limited .NET Framework libraries and its XAML parser as compared to the full WPF platform.

The initial versions of Silverlight (Silverlight 1.0 and 2.0 versions) were mainly targeted toward building media applications, so it focused on media integration, vector graphics, and animation. Later versions (Silverlight 3 and Silverlight 4) enhanced and streamlined Silverlight's media applications capabilities, and extended the product focus to implementing data-driven enterprise line of business (LoB) applications. Silverlight 5 version extended the LoB applications capabilities and added support for mobile applications development, as well as support for gaming and 3-D animations.