Rob Eisenberg
Christopher Bennage

Full Color
Code samples appear as they do in Visual Studio!

Sams Teach Yourself WPF in 24 Hours

SAMS

Sams Teach Yourself WPF in 24 Hours

Copyright © 2009 by Pearson Education, Inc.

All rights reserved. No part of this book shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without written permission from the publisher. No patent liability is assumed with respect to the use of the information contained herein. Although every precaution has been taken in the preparation of this book, the publisher and author assume no responsibility for errors or omissions. Nor is any liability assumed for damages resulting from the use of the information contained herein.

ISBN-13: 978-0-672-32985-2
ISBN-10: 0-672-32985-9

Library of Congress Cataloging-in-Publication Data

Eisenberg, Robert (Robert Harold)

Sams teach yourself WPF in 24 hours / Robert Eisenberg and Christopher Bennage.

p. cm.

ISBN-13: 978-0-672-32985-2

ISBN-10: 0-672-32985-9

1. Windows presentation foundation. 2. Application software. 3. Windows (Computer programs)—Standards. 4. Microsoft .NET. I. Bennage, Christopher. II. Title. III. Title: Teach yourself WPF in 24 hours.

QA76.76.A65E39 2009

006.7'882—dc22

2008020014

Printed in the United States of America

First Printing July 2008

Trademarks

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Sams Publishing cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

Warning and Disclaimer

Every effort has been made to make this book as complete and accurate as possible, but no warranty or fitness is implied. The information provided is on an “as is” basis. The authors and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book or from the use of the programs accompanying it.

Bulk Sales

Sams Publishing offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales. For more information, please contact

U.S. Corporate and Government Sales
1-800-382-3419
corpsales@pearsontechgroup.com

For sales outside the U.S., please contact

International Sales
international@pearson.com

Editor-in-Chief
Karen Gettman

Executive Editor
Neil Rowe

Development Editor
Mark Renfrow

Managing Editor
Kristy Hart

Project Editor
Betsy Harris

Copy Editor
Barbara Hacha

Indexer
Brad Herriman

Proofreader
Debbie Williams

Technical Editor
J. Boyd Nolan

Publishing Coordinator
Cindy Teeters

Book Designer
Gary Adair

Composition
Nonie Ratcliff

This Book Is Safari Enabled

The Safari® Enabled icon on the cover of your favorite technology book means the book is available through Safari Bookshelf. When you buy this book, you get free access to the online edition for 45 days.

Safari Bookshelf is an electronic reference library that lets you easily search thousands of technical books, find code samples, download chapters, and access technical information whenever and wherever you need it.

To gain 45-day Safari Enabled access to this book

- Go to www.informit.com/onlineedition.
- Complete the brief registration form.
- Enter the coupon code F613-Z84G-8HWN-PHGQ-1G5E.

If you have difficulty registering on Safari Bookshelf or accessing the online edition, please email customer-service@safaribooksonline.com.

Introduction

Windows Presentation Foundation, or WPF, is Microsoft's latest framework for building sophisticated and rich user interfaces for desktop applications. WPF differs significantly from its predecessor, and yet draws on many of the concepts found existing in frameworks for both desktops and the web.

WPF enables developers to easily and quickly handle tasks that were either very difficult or impossible to accomplish in previous frameworks.

Audience and Organization

This book is intended for those who have at least some experience with general .NET development. If you have worked with WinForms or ASP.NET, you should feel comfortable with this book. The code examples provided are written in C#, but we've been careful to keep them readable for those whose primary language is Visual Basic.

Because WPF is both a broad and a deep topic, it can easily become overwhelming. Our approach in this book is to stay broad. We cover the essential concepts of the framework. Our goal is for you to feel confident building a WPF application when you are done with the book, as well as equipping you to dig deeper into any areas of the framework that interest you.

The book is organized into five parts. In each of the first four parts, we build a sample application that demonstrates the features of WPF covered in that part. Although the applications are simplified, they are designed to reflect real-world scenarios that you are likely to encounter. Each of the parts builds on its predecessor, and we recommend reading them in order. Part V concludes with information designed to help you move forward after the book.

- **Part I, "Getting Started"**—We build a utility for browsing the fonts installed on your system. You'll learn about the new markup language XAML that is an integral part of WPF. We also introduce you to most of the basic controls, including those that handle layout. You'll also learn about basic data binding in WPF.
- **Part II, "Reaching the User"**—You'll create your own rich text editor. You'll learn about the powerful new event and command systems. We also introduce you to a few more controls and show you how you can deploy your WPF applications. You also discover how to print from WPF.

- **Part III, “Visualizing Data”**—This part teaches you how to style an application, as well as how to use WPF’s powerful graphics capabilities for visualizing the data in your applications. We also dig further into data binding and show you some options for architecting your WPF applications.
- **Part IV, “Creating Rich Experiences”**—You’ll learn how to easily embed media in your applications. You’ll see how WPF’s drawing and templating APIs make it easy to create unique and visually attractive interfaces. You’ll also get started with animation.
- **Part V, “Appendices”**—This includes a brief introduction to 3D and a list of tools, frameworks, and other resources that aid in WPF development.

Throughout the book, we use code-continuation characters: When a line of code is too long to fit on the printed page, we wrap it to the next line and precede it with a code-continuation character, like this:

```
public object ConvertBack(object value, Type targetType, object parameter,
➥CultureInfo culture)
```

Farther Up and Further In

Learning WPF is really a lot of fun. We’ve discovered a new joy in building user interfaces since we’ve begun using this technology. We believe that you’ll have the same experience working through this book. Although it may take some time to become a master of WPF, it’s actually quite easy to get up and running quickly. By the time you are done here, you’ll be ready to start using WPF on your next project.

Now, let’s get started!

HOUR 1

What WPF Is and Isn't

What You'll Learn in This Hour:

- What WPF is
- When you should use WPF
- What tools you will need
- How WPF compares to other frameworks
- The versions of .NET
- Silverlight

What Is WPF?

WPF is big. In fact, it can be overwhelming because it has lots of moving parts that all interconnect. The shortest answer to the question, though, is that WPF is an API for building graphical user interfaces (UI) for desktop applications with the .NET Framework.

Now for the longer answer.

To begin with, WPF is an abbreviation for *Windows Presentation Foundation*. Physically, it's a set of .NET assemblies and supporting tools. It's intended to provide a unified API for creating rich, sophisticated user interfaces on Windows XP and Windows Vista.

WPF combines the good things from web development, such as style sheets and a markup language for declarative UI, with good things from Rich Internet Applications, such as scalable vector graphics, animation, and media support. These good things are wrapped up with the good things from traditional Windows development—things like strong integration with the OS and data binding. In WPF, these concepts are strengthened and unified. Even all that does not capture the full extent of WPF. It has other facets, such as support for 3D drawing, advanced typography, and portable documents similar to PDF.

WPF is also a *unified* API. Many of the things you are able to do in WPF, you could do before. However, doing them all in one application was extremely difficult. Not only does WPF enable you to bring these disparate features together, but it provides you with a consistent API for doing so.

WPF is just one part of a larger picture. Three additional libraries were also released as part of .NET 3.0. All four of these libraries have the same intent of providing a consistent, unified API for their domain. Additionally, combining any of these four libraries in an application can yield some very impressive results. The three sibling libraries of WPF are shown in Table 1.1.

TABLE 1.1 The Sibling Libraries of WPF

WCF	Windows Communication Foundation is focused on messaging. This API greatly simplifies all sorts of networking and communication tasks. It covers everything from web services to remoting to P2P and more.
WF	A powerful library for building workflow enabled applications. It utilizes a markup language for declaring workflows in an application, and thus prevents workflow from becoming hard-coded. It also makes it very easy for developers to create custom workflow tasks.
CardSpace	The least famous of the four libraries, CardSpace provides a common identification system that can be used by desktop applications, web sites, and more.

The immediate predecessor to WPF is Windows Forms, the graphical API available to developers in .NET 2.0 and earlier. Windows Forms provides a managed wrapper for accessing the graphical functions of the traditional Windows API. WPF differs fundamentally in that it builds on top of DirectX. The DirectX API was originally focused on multimedia and game programming in particular. As such, you are able to do some nifty visual tricks in WPF that were practically impossible with Windows Forms. It also means that WPF will take advantage of hardware acceleration when it is available.

WPF still has some similarities to Windows Forms (and even ASP.NET Web Forms). Microsoft provides a library of basic controls such as text boxes and buttons. You'll also encounter familiar concepts such as data binding and code-behind files. All these concepts have been refined and improved for WPF.

Getting to Know the Features of WPF

Let’s take a moment to review the major features of WPF. We’ll cover each of these with more depth in later hours.

Declarative UI

WPF allows you to construct your interface using a markup language called XAML (pronounced *zammel*, rhymes with *camel*). We’ll dig into XAML in Hour 2, “Understanding XAML,” but if you have ever worked with HTML, you are already familiar with the concepts. XAML is a much richer markup language than HTML, and it has less ambiguity. Visual Studio, as well as some members of the Expression family of products are able to generate XAML natively.

XAML provides a common medium for interacting with designers.

Intelligent Layout

Arranging the various components of an application onscreen can be complicated, and it’s further complicated by the myriad display possibilities that users might have. WPF provides an extensible layout system for visually arranging the elements of a user interface. It can intelligently resize and adjust, depending on how you define the layout. We’ll cover this in some detail when we discuss panels in Hour 4, “Handling Application Layout.”

Scalable Graphics

Graphics in WPF are vector based, in contrast to raster based. Vector graphics are inherently scalable and typically require less storage than a comparable raster image. WPF still has plenty of support for raster graphics, but vectors are an excellent fit for constructing user interfaces.

Vector graphics have already become popular on the web, primarily because of Adobe Flash and to a lesser extent the Scalable Vector Graphics specification (SVG).

The net result for developers with WPF is that applications scale nicely without a loss in visual quality.

By the Way

Vector Versus Raster

A raster graphic is an image that is stored as rectangle grid of pixels, and each pixel is assigned a color. Most graphic file formats that you are familiar with are just variations to this method. This includes formats such as GIF, JPEG, BMP, and PNG.

Raster graphics are also called *bitmaps*. (Don't let the the BMP file format confuse you. The term *bitmap* is a general term describing a particular way to store image data.)

Suppose that you have a raster image of a blue circle on a white background that is 100×100 pixels. The computer loads those 10,000 pixels into memory and displays them on the screen. That's a lot of data for such a simple image. Imagine that we need the same image but two or three times larger. The number of pixels increases exponentially. If we could simply provide the computer with the dimensions, position, and color of the shapes, then we would have much less data to worry about. In this way, raster graphics are inefficient.

Another problem with raster images is that they do not resize well. There's a noticeable loss of quality, especially when you are enlarging an image. Suppose that you wanted to double the size of a 100×100 image of yourself. To increase the size to 200×200, you would need 390,000 more pixels. These missing pixels would need to be interpolated from the existing ones.

Vector graphics, however, are stored as geometries. The data structure for a vector image contains just enough information for the computer to draw the image. A vector image of a blue circle on a white background would contain the x and y position of the circle, its radius, and metadata indicating the circle was blue and the background white. When a computer renders this image, it figures out the actual pixels on-the-fly. This means that there is no difference in quality between the 100×100 vector image and the 200×200 image, and that the size of the data needed to draw the image is substantially less.

A general rule of thumb is that vector graphics are good for geometrical or cartoonish images and that raster is better for photographs and realistic images.

Templates

WPF makes it very easy to create reusable elements for your user interfaces. There are two types of templates in WPF: control templates and data templates. Control templates enable you to redefine the way a control looks. (For ASP.NET developers, they are conceptually similar to control adapters.) For example, if your application needs to have all its list boxes with a blue background and a red border, you could use a control template to redefine the visual appearance of list boxes. Control templates also make it easier for designers. They are able to provide a "look" for a list box through a control template, with little to no impact on the actual development process.

Data templates are similar, except that instead of defining the way a control looks, they define the way certain types of data are rendered. Imagine that you have an application dealing with people, such as a contact manager, and that you represent people in code with instances of a `Person` class. You can create a data template that defines how an instance of a `Person` is rendered in the UI. For example, an instance of `Person` might be visualized as a business card with a picture, first name, last name, and telephone number. If you use such a data template, whenever a `Person` instance is bound to some UI element, such as a list box, WPF will use the corresponding data templates. In practice you will find that data templates are really handy when dealing with lists or other collections of data.

Binding

When we talk about binding in WPF, you probably jump immediately to the concept of *data binding*. Data binding has already been made popular with Windows Forms and ASP.NET Web Forms, and has demonstrated its usefulness there. Although WPF has significant data binding features—significant in that it greatly outclasses its predecessors—it also allows you to declaratively bind other things such as commands, key bindings, animation, and events. For example, you can declaratively bind a button control to a command for pasting.

Styling

WPF really shines when it comes to making an application look pretty. It allows you to do such things as make the background of a text box red or surround a button with a thick blue border. Styles in WPF are similar to cascading style sheets for HTML. Though again, WPF styles are richer and have less ambiguity. They encompass all the visual characteristics you would expect, such as padding, margin, position, color, and so on. But you can also use styles to declare nonvisual properties.

Styles are also easy to reuse, and when you combine them with templates, you are able to do some amazing things.

Triggers

Both templates and styles in WPF support the notion of triggers. A trigger enables you to tell WPF something like this: “When the mouse is over the button, make the background purple.” In other words, triggers enable you to declaratively handle changes of state. You will also find them useful for kicking off animations.

Animation

The animation framework in WPF is very impressive, and a great deal more useful than you might think. Most properties in WPF can be animated, and support exists for timelines, key frames, and interpolation. Animations are easily integrated with templates and styles. For example, you might define a style for a button that animates the button when you move the mouse over it. Flash developers and designers will be impressed with the available features.

3D

Finally, WPF allows for some basic 3D modeling and animation. I say basic because WPF is not intended for building high-performance 3D applications. You won't be constructing a first person shooter in WPF. (If that is what you are interested in, be sure to give Microsoft's XNA platform a look.) Nevertheless, the 3D features are powerful and easily integrated into any user interface. We won't be covering the 3D features of WPF in this book; however, a very basic tutorial is available in the appendixes.

Why Use WPF?

WPF, as well as its sister libraries released with .NET 3.0, are well-factored and consistent APIs. They unify many programming concepts and, on the whole, make a lot of complicated development tasks easier. However, WPF is not necessarily the right choice for every project. Some desktop applications would be easier to build and maintain in Windows Forms. But, you'll find many benefits when you work with WPF. Any Windows developer should begin learning WPF because it will eventually mature to a point where it completely replaces Windows Forms.

Many of the key benefits are apparent by reading the list of features in the "Getting to Know the Features of WPF" section. The following are some scenarios where WPF will really shine:

- Your project requires collaboration with designers. The use of XAML and its supporting tools can really help out here. After the developers and the designers become familiar with the tools, your team can experience tremendous gains in efficiency.
- Your application is media aware. If you need to integrate video and audio into your product, you'll definitely want to consider WPF.

- The anticipated hardware for your application has support for DirectX 9 or greater. WPF is built on top of DirectX, and your applications will benefit from the hardware acceleration.
- Your application needs support for advanced typography. WPF has support for OpenType and many other features that are not available with Windows Forms.

Finally, you as a developer can get more done in less time. Even if you are not concerned with many of the bells and whistles of WPF, you will be able to produce quality software with less effort. In Part I, “Getting Started,” we’ll demonstrate this principle by building a simple but useful utility using just markup language.

Comparing WPF to Other Options

If you are solely a .NET developer, you really have only two other options to consider: Windows Forms and ASP.NET. We’ve already compared WPF to Windows Forms throughout the course of this hour. The only real advantages that Windows Forms has are its mature library of controls and significant third-party support. WPF is still the new kid on the block, and the mass of supporting tools and materials has not had time to build up yet.

Comparing WPF to ASP.NET is a little more involved. The question here really centers on deployment and distribution. WPF is currently limited to the Windows platform, and there’s obviously no such limitation with a web application. WPF requires the .NET Framework 3.0 or later, as well as a means of deploying the application. If your application is centralized, requiring one or more server components, you are likely to reduce the complexity significantly by choosing to develop a web application.

Outside of the .NET world, some of the same features are available with Adobe Flash, primarily when it comes to media and animation. Historically, Flash has really only been useful in a Web context. However, the Adobe AIR platform utilizes Flash for developing cross-platform, desktop applications. Nevertheless, Flash still has some notable drawbacks. The development environment is not as robust as .NET although, admittedly, Flash does tend to be more designer friendly. Control libraries for Flash are much more limited and cumbersome to use. It is possible that AIR will provide some healthy competition for WPF.

The Pieces of .NET Framework

Unfortunately, a lot of terms and version numbers are floating around in the .NET world right now, and sorting them out can be particularly difficult. Let's take a moment and step through the various pieces of the .NET Framework and how they relate to all the version numbers.

It is easiest to think of the .NET Framework as a family of products including the runtime, the compilers, and the common code libraries.

The runtime is the common language runtime, or CLR. It is the virtual machine that hosts .NET applications. It provides many of the core services such as memory management, garbage collection, and security. It's outside the scope of this book to discuss the CLR in depth, but you should know that the CLR is the .NET runtime and its version numbers differ from those of the .NET Framework in general. The current CLR is 2.0.

The two dominant languages in the .NET world are C# and Visual Basic .NET. Both of these languages have their own version numbers and those numbers differ from the .NET Framework as a whole. The current version of C# is 3.0, and Visual Basic is 9.0.

You'll also hear about the Base Class Library (BCL) and the Framework Class Library (FCL). The BCL is a set of classes available to any language in the .NET family. These classes mostly live in the `System` namespace. The FCL is a term that includes both the BCL and the common libraries in the `Microsoft` namespace. Distinguishing between the two sometimes results in hair splitting, and many people use the terms interchangeably.

Figure 1.1 shows how these "products" have changed with each release of the .NET Framework beginning with 2.0.

Some interesting points to clarify are the following:

- The CLR has not changed since the release of 2.0. Thus, the core features of the .NET Framework are the same.
- C# 3.0 and VB .NET 9.0 both compile to bytecode (or IL) that is compiled "just in time" on the CLR 2.0. The new language features with .NET 3.5 are essentially enhancements to the respective compilers.
- WPF is a class library; nothing changed the underlying CLR. This means that unlike .NET 2.0, version 3.0 of the Framework was just the addition of new libraries.

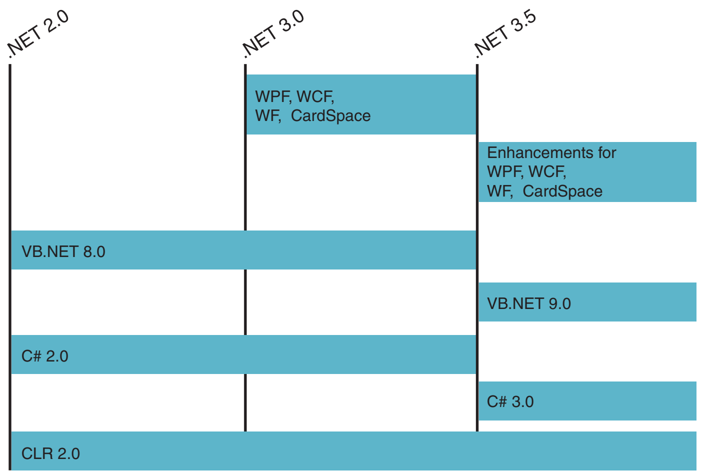

FIGURE 1.1 The version history of the .NET Framework.

Tools for WPF

In this book we work primarily with Visual Studio 2008. Specifically, we use the Express Edition, which Microsoft provides free of charge. Visual Studio 2008 has native support for WPF applications.

By the Way

The Express Edition of Visual Studio 2008 is available at www.microsoft.com/express/, along with many other resources.

It is possible to build WPF applications with Visual Studio 2005; however, you need to install the WPF extensions for Visual Studio that never made their way to a final release. I strongly advise to move to 2008 if at all possible.

You can also use SharpDevelop (also known as #develop). It is an open-source IDE for .NET, and it has support for building WPF applications in .NET 3.0. It is a solid IDE, but it is hard to beat the level of support for WPF in Visual Studio.

The second primary tool for creating WPF applications from Microsoft is Expression Blend. Blend targets designers rather than developers. It works with the same files as Visual Studio, so a designer using Blend and a developer using Visual Studio can both work on the same projects, solution, and files. Blend is somewhat comparable to the IDE for Adobe Flash. You will find drawing tools, animation timelines, palettes, and other designer-centric features. Despite its focus, I recommend that developers become familiar with Blend. Blend is also one of the first Microsoft products to be written with WPF.

A third-party product exists for designing WPF interfaces—Aurora, by Mobiform Software. It provides a similar set of features as Expression Blend. One notable feature is that Aurora designer can be embedded in another WPF application. So if you have a need for providing a built-in XAML editor in your application, be sure to check it out.

Expression Design is another Microsoft product. It is for authoring vector-based graphical content, similar to Adobe Illustrator or Inkscape. Expression Design can be used to create logos, icons, and illustrations for use with WPF. It can natively output graphics as XAML, which can then be directly incorporated into WPF. Expression Design differs from Blend, in that Blend's focus is purely on authoring user interfaces.

Many other applications for producing 2D and 3D art now have plug-ins available for exporting assets in XAML. (Remember, XAML is the native tongue of WPF.) Some of the applications that have plug-ins available are Adobe Fireworks, Adobe Illustrator, Inkscape, Maya, Blender, and Lightwave.

Aside from the 3D tools just mentioned, at least one WPF-specific 3D content editor is available—ZAM 3D by Electric Rain. ZAM 3D is very similar to the Swift 3D product for Flash. It's more approachable than most 3D editors and is probably the best place to start for WPF developers interested in 3D.

One final tool worth mentioning is Kaxaml. It is a lightweight XAML editor featuring a live preview. That is, you can see how WPF will render your markup as you are typing. It is a very handy utility to have around, and at the time of this writing it is free.

By the Way

Visit www.kaxaml.com/ to download Kaxaml. Even though the tutorials in this book focus on Visual Studio, you might find it useful to test some of the markup in Kaxaml. Unlike the preview in Visual Studio, Kaxaml is truly what-you-see-is-what-you-get (WYSIWYG).

Many other tools, utilities, control libraries, and so on become available every day. Some are third-party commercial products, and others are community-driven and free. Be sure to check in the appendixes for additional resources. For the sake of simplicity, we use only Visual Studio 2008 Express Edition in this book.

Constrasting WPF with Silverlight

Silverlight is a platform for developing Rich Internet Applications (RIA), whereas WPF is primarily intended for desktop applications. Silverlight is a direct competitor to Adobe Flash and it has a strong focus on media, cross-platform compatibility, as well as a small download and install footprint. Like Flash, Silverlight applications are hosted in a browser.

Microsoft has intentionally designed Silverlight to be very similar to WPF, although the two are separate products. In fact, an early name for Silverlight was WPF/E or Windows Presentation Foundation/Everywhere. Developers familiar with one technology will have a head start with the other.

Like WPF, Silverlight uses XAML for declaring user interfaces. In version 1.0, a Silverlight application consists only of text files containing JavaScript and XAML. Silverlight 2.0, however, will support a more robust runtime and a base class library similar to the standard .NET BCL. You will be able to write Silverlight applications in your favorite .NET language and compile them to assemblies for distribution. Silverlight 2.0 will look a lot more like WPF, but you should be aware that significant differences exist. It is almost certain that Silverlight will not support all the features in WPF. Likewise, code written for Silverlight may need significant changes before it will compile for a standard .NET application. Always keep in mind that the runtime for Silverlight is different from the CLR.

Summary

Windows Presentation Foundation is the future of software development for desktop applications. The API is very large, and the features are numerous. Becoming an expert in WPF can take some time. However, a basic understanding of the core features can greatly increase a developer's productivity. WPF is also a leap forward in promoting collaboration between designers and developers.

Q&A

Q. ***Are there any good reasons for not using WPF to build your application?***

A. A WPF application is generally more resource intensive than a Windows Forms application. If you are building applications for low-end hardware, you might want to do some performance testing before you commit to using WPF. Additionally, .NET 3.0 and 3.5 are not yet widely installed and they are prerequisites for a WPF application. (.NET 3.0 is included with Vista and Windows Server 2008.)

Q. ***There seems to be a lot to understanding WPF; do I really need to master all of these concepts to use it?***

A. No, you don't. As we've emphasized, WPF is big. However, by the end of Part I, you can begin building useful applications and realizing the benefits of WPF.

Workshop

Quiz

1. What is the benefit of using a markup language for designing a user interface?
2. What operating systems does WPF currently support?
3. How does WPF differ from Silverlight, and in what ways are they similar?
4. Aside from Visual Studio 2008, what is another tool from Microsoft that WPF developers should become familiar with?

Answers

1. Using a markup language such as XAML, or even HTML, is beneficial because it provides a common medium for both designers and developers. Additionally, the markup language allows for a declarative approach for building applications, which is often easier to construct and maintain.
2. WPF is currently available on Windows XP and Windows Vista.
3. WPF is part of the .NET Framework, and it is intended for building graphical user interfaces for desktop applications. Silverlight targets Rich Internet Applications that are hosted in a web browser. Silverlight's runtime is different from the standard .NET runtime. Both Silverlight and WPF use XAML for defining interfaces. Microsoft has intentionally made Silverlight as close to WPF as possible to lower the barrier for developers and designers.
4. WPF developers should be at least somewhat familiar with Microsoft's Expression Blend. The application is primarily intended for designers, but can often be useful for developers as well. It uses the same file formats as Visual Studio, so that solutions and projects are interchangeable between the two applications.

Index

Symbols

3D animation, 10

3D modeling, 10

3DS, 425

A

abstracting data stores, 188, 190

acceleration, animations, 392-393

AccelerationRatio property, 392

accessing controls, labels, 64-67

actions, triggering, buttons, 67-68

Adding a ToolTip to the Font Viewer listing (6.2), 83-84

Address.cs listing (12.2), 183-187

Adobe Fireworks, 424

Adobe Illustrator, 424

Advanced Grid Layout listing (4.2), 50-51

aggregates, 187

Alignment function, 42-44

analogous colors, 317

animation, 383
- acceleration, 392-393
- adding, EventTrigger, 377-379
- clock animations, 400
- clocks, 386-388
- controlling, 398-400
- deceleration, 392-393
- fills, 393
- frames, 383
  - frame rate, 383
  - key frames, 384, 394-396
  - tweens, 384

local animations, 400
manual animations, 401
Media Viewer, 388-390
applying to, 401-405
motion, 383
path geometry, 396-398
placement, 391-392
properties, 392-394
repeating, 393
target property, resolving, 390-391
timelines, 384-388
animation framework, 10
Animation Toolbar (ZAM 3D), 427
anticipating user needs, 131
APIs
DirectX API, 6
unified APIs, 6
App.xaml listing (14.1), 212
App.xaml.cs listing (11.5), 169-170
application frameworks, 425
Application Presenter
implementing, 196, 199
application shell, creating, 179-182
ApplicationCommands built-in command library, 146
ApplicationController.cs listing (17.1), 269-270
ApplicationPresenter.cs listing (13.2), 197-199
applications
animations
acceleration, 392-393
adding, 377-379
applying to, 401-405
clock animations, 400
controlling, 398-400
deceleration, 392-393
fills, 393
key frames, 394-396
local animations, 400
manual animations, 401
path geometry, 396-398
placing, 391-392
properties, 392-394
repeating, 393
bitmaps, outputting, 168-172
colors, choosing, 316-318
commands, 145-146
adding to, 151-152
availability, 153
binding to menus, 148-149
customizing, 149-151
input bindings, 153-155
using, 146-147
control templates, 347-350
adding to, 353-358
benefits, 366
controlling, 351-352
creating, 362-366
creating for a slider, 360-362
identifying special parts, 358-362
items controls, 351-352
placing content in, 350-351
deployment models, 93-98
designing, 407, 418
architecture concerns, 412-414
code conventions, 417-418
code organization, 413
coding conventions, 417-418
dependency injection, 410-411
DRY (Don’t Repeat Yourself), 409, 412
inversion of control, 410-411
MVP (Model View Presenter), 413
orthogonal code, 408, 411
patterns, 409-412
resources, 415
SOC (Separation of Concerns), 408, 412
SRP (Single Responsibility Principal), 408, 412
styles, 415-416

templates, 416-417
UI design, 412-414
UI layout, 414-415
usability, 414
XAML, 418
YAGNI (Ya Aren't Gonna Need It), 411
developing, 33-39
Edit Contact screens, building, 201-207
events
handling, 128-138
preview events, 139-141
file formats, 93
Font Viewer, fancy instructions, 61
formatted text support, adding to, 115, 117-118
Front Builder, building, 27-39
Front Viewer, laying out, 41-57
"hello world" XAML applications, creating, 18
layout, 41, 57
Alignment function, 42-44
building up, 56
Canvas function, 53-54
creating, 108-109
decorators, 41-42
Decorators function, 54-56
DockPanel function, 46-48
Grid function, 48-52
Margin function, 42-44
panels, 41-42
Panels function, 45
StackPanel function, 45-47
System.Windows.FrameworkElement base class, 42
WrapPanel function, 52-54
media player
creating, 282-285
finalizing, 286-288
menu screen, implementing, 271-274
menus, adding to, 113-115
models
building, 274-277
defining, 182-187
navigation applications, 99-100
navigation models, 98-100
picture screen, creating, 277-281
Print Manager, developing, 161-164
Print Preview, adding to, 164-168
requirements, defining, 267
RIAs (Rich Internet Applications), 14
shells, enhancing, 200-201
solutions, setting up, 268-271
text document editor, 107
formatted text support, 115-118
layout, 108-109
menus, 113-115
toolbars, 109-110
user controls, 110-113
toolbars, adding to, 109-110
triggers
advanced scenarios, 380-381
DataTrigger, 373-377
EventTrigger, 377-379
leveraging, 369-373
restrictions, 380
user controls, 100-104
adding to, 110-113
XAML file, converting to, 94-95
XBAPs
converting to, 96-97
publishing, 97-98
applying data templates, 259
architectures
application design, 412-414
Contact Manager, choosing, 178-179
ASP.NET, 1
WPF, compared, 11

Aurora, 14, 424

automatic change notifications, demonstrating, 85-88

AutoReverse property, 392

availability, commands, determining, 153

B

Base Class Library (BCL), 12

BasedOn property, 227

BCL (Base Class Library), 12

BeginStoryboard inheritor (TriggerAction), 379

BeginTime property, 392

BevelBitmapEffect, 342

binding, 9, 75, 89, 229-232

- automatic change notifications, 85-88
- bound data, formatting, 239-247
- change, communicating with, 234-239
- contact lists, adding, 230-232
- in code, 232-234
- syntax, 76, 89
- two controls, 76-77
- two-way data binding, 78-85

Binding a Slider to a TextBlock listing (6.1), 77

Binding markup extension, 23

bindings, input bindings, creating, 153-155

bitmap effects

- BevelBitmapEffect, 342
- BitmapEffectGroup, 342
- BlurBitmapEffect, 342
- creating, 339-341
- DropShadowBitmapEffect, 341
- EmbossBitmapEffect, 342
- Media Player, adding to, 342-344
- OuterGlowBitmapEffect, 341

BitmapEffectGroup, 342

bitmaps, 8

- outputting, 168-172

Blend, 13, 424

Blender, 424

BlurBitmapEffect, 342

“boiler-plate” code, 425

Bold inline element (TextBlock), 61

bound data, formatting, 239-247

brushes, 318-319, 328

- Contact Manager, creating for, 213
- DrawingBrush, 323-325
- LinearGradientBrush, 319-320
- RadialGradientBrush, 321-323
- SolidColorBrush, 319
- VisualBrush, 326-327

bubbling up events, 122

built-in command libraries, 146

Button, 67-68

Button class, Name property, 20

buttons, 67

- actions, triggering, 67-68
- toggling, 68-71

C

C# 3.0, 12

CAB (Composite Application Block), 425

Caliburn, 425

Canvas function, 53-54

CardSpace library, 6

carrot animation, creating, ZAM 3D, 428-435

change notifications, 85

changes, data bindings, communicating with, 234-239

child elements, 21

choosing panels, items controls, 351-352

CI (constructor injection), 199

classes

- Application Presenter class, implementing, 196, 199
- Button class, Name property, 20
- Clock, 386-388

content-related classes, 252
DataTemplateSelector, 255
hierarchy, 171
ItemContainerGenerator, 254
list-related classes, 252
MessageBox class, 71
partial classes, 30
PresenterBase.cs class, creating, 193-194
repository classes, 188
Selector, 260-261
SideBar.xaml.cs, 200
StyleSelector, 255
System.Windows.Controls. Control base class, 42
System.Windows.Framework Element base class, 42

clock animations, 400

Clock class, 386-388

clocks, animation, 386-388

CloseTab method, 199

CLR (common language runtime), 12, 122

code
application design, conventions, 417-418
“boiler plate” code, 425
data binding in, 232-234
downloading, 437
organizing, 413
orthogonal code, application design, 408, 411
refactoring, 32

code-behind files (Visual Studio 2008), 29-30

Code-Behind for MainWindow.xaml listing (9.3), 138

Code-Behind for TextEditorToolbar.xaml listing (9.2), 136-138

collection views, 247-248
XAML, 248-249

colors, 315-316, 328
analogous colors, 317
brushes, 318-319
DrawingBrush, 323-325
LinearGradientBrush, 319-320
RadialGradientBrush, 321-323
SolidColorBrush, 319
VisualBrush, 326-327
choosing, 316-318
complementary colors, 317
declaring, 212
primary colors, 317
secondary colors, 317
tertiary colors, 317

combining resources, 213

ComboBox named part (controls), 358

commands, 145-146
availability, determining, 153
built-in command libraries, 146
customizing, 149-151
DeleteContact, 179
gestures, defining, 155
input bindings, creating, 153-155
menus, binding to, 148-149
Refactor menu, Rename, 31-32
SaveContact, 179
Search, 179
Text Editor, adding to, 151-152
using, 146-147

common language runtime (CLR), 122

communicating changes to data bindings, 234-239

compiling, Contact Manager, 207

complementary colors, 317

complex drawings
path geometry, 306-310
stream geometry, 310-311

ComponentCommands built-in command library, 146

Composite Application Block (CAB), 425

constructor injection (CI), 199

contact forms, 64, 68-70

contact lists, Contact Manager, adding to, 230-232

Contact Manager, 177, 190, 193, 206-207, 249
Application Presenter, implementing, 196, 199
application shell, creating, 179-182

architecture, choosing, 178-179
brushes, creating for, 213
code, downloading, 437
collection views, 247-248
    XAML, 248-249
compiling, 207
contact lists, adding, 230-232
data binding, 229-232
    code, 232-234
    communicating change with, 234-239
    formatting bound data, 239-247
data store, abstracting, 188, 190
data templates, 237-239
designing, 177-178
Edit Contact screen, building, 201-207
items, displaying, 253
ItemsControl, 262
    Selector class, 260-261
List View, displaying, 236-237
lists
    control hierarchy, 251-252
    ItemsControl, 252-256
models, defining, 182-187
PresenterBase.cs class, creating, 193-194
resource files, factoring, 216
resources
    combining, 213
    defining, 211-212
    using UIs in, 214-215
SearchBar, adding to, 214-215
shell, enhancing, 200-201
side bars, customizing, 256, 258-259
solution shell, creating, 179-182
styles, 227
    defining, 217-220
    factoring, 226-227
    keyed styles, 220-221, 226
tabbed UI infrastructure, completing, 194-196

Contact.cs listing (12.3), 184-187

ContactRepository.cs listing (12.5), 188-190

contacts, aggregates, 187

Content property, 21-22

content-related classes, 252

control, 42

control hierarchy, lists, 251-252

Control Manager, lists, laying out, 256

control templates, 347-350
applications, adding to, 353-358
benefits, 366
content, placing in, 350-351
controlling, 351-352
creating, 362-366
items control, choosing panels in, 351-352
sliders, creating for, 360-362
special parts, identifying, 358-362

controlling
animations, 398-400
control template, 351-352

controls, 41, 72. *See also* functions
accessing labels, 64-67
binding, 76-77
    two-way data binding, 78-85
buttons, 67
    toggling, 68-71
    triggering actions, 67-68
control templates, 347-350
    adding to applications, 353-358
    benefits, 366
    controlling, 351-352
    creating, 362-366
    creating for a slider, 360-362
    identifying special parts, 358-362
    items controls, 351-352
    placing content in, 350-351
DataContext, 80-83

document controls, 168
leveraging, 59
lists, displaying, 71
media controls, styling, 303-306
named parts, 358-360
RichTextBox, 115, 117-118
ScrollViewer, 201
user controls, 100-104, 110-113

conventions, code, application design, 417-418

ConvertBack method, 240, 244-245

Converter in Application Resources listing (15.1), 243

ConverterParameter, 246

converters
bound data, formatting, 239-247
hooking up, 242-244
parameterizing, 246-247

converting applications
to XAML files, 94-95
to XBAPs, 96-97

custom controls, user controls, compared, 101

custom dependency properties, 167

custom document paginators, implementing, 158-160

Custom DocumentPaginator listing (11.1), 158-160

customizing commands, 149-151

D

data binding, 9, 75, 89, 229-232
automatic change notifications, 85-88
bound data, formatting, 239-247
change, communicating with, 234-239
contact lists, adding, 230-232
in code, 232-234
modes, 79
syntax, 76, 89
two controls, 76-77
two-way data binding, 78-85

data stores, abstracting, 188, 190

data templates, 237-239
applying, 259

DataContext property, 80-83, 199

DataTemplateSelector class, 255

DataTrigger, 373-377

deceleration, animations, 392-393

DecelerationRatio property, 392

declarative UI, 7

declaring colors, 212

decorators (layout), 41-42
UIs, enhancing with, 54-56

DefaultStyles.xaml listing (14.4), 217-218

defining
gestures, commands, 155
models, 182-187
requirements, applications, 267
resources, 211-212
styles, 217-220

DeleteContact command, 179

demonstrating automatic change notifications, 85-88

dependency injection (DI), 199
application design, 410-411

dependency properties, 84-85, 234
custom dependency properties, 167

deployment models, applications, 93-98

designer tools, 424

designing
applications, 407, 418
architecture concerns, 412-414
code organization, 413
coding conventions, 417-418
dependency injection, 410-411
DRY (Don’t Repeat Yourself), 409, 412
inversion of control, 410-411
MVP (Model View Presenter), 413

orthogonal code, 408, 411
patterns, 409-412
resources, 415
SoC (Separation of Concerns), 408, 412
SRP (Single Responsibility Principal), 408, 412
styles, 415-416
templates, 416-417
UI design, 412-414
UI layout, 414-415
usability, 414
XAML, 418
YAGNI (Ya Aren't Gonna Need It), 411
Contact Manager, 177-178
developer tools, 423
developing applications, 33-39
DI (dependency injection), 199
dialog boxes, modal dialog boxes, 71
DirectX API, 6
discrete interpolation, key frames, 395
Dispatcher, 277
displaying
items, 253
List View, 236-237
lists, 71
rich text, 60-61
DockPanel function, 46-48
document controls, 168
document paginators, implementing, 158, 160
DocumentPageView document control, 168
DocumentViewer document control, 168
DocumentViewer named part (controls), 358
***Domain-Driven Design: Tackling Complexity in the Heart of Software*, 191**
downloading code, 437
drawing
ellipses, 299
fills, 295-297
headers, Media Viewer, 299-302
lines, 292-295
path geometry, 306-310
polygons, 296-298
polylines, 295-298
rectangles, 299
stream geometry, 310-311
strokes, 292-295
DrawingBrush, 323-325
DropShadowBitmapEffect, 341
DRY (Don't Repeat Yourself) principal, application design, 409, 412
DynamicResource markup extension, 23
StaticResource markup extension, compared, 216

E

Edit Contact screens, building, 201-207
EditContactPresenter.cs listing (13.4), 204-206
EditContactView.xaml listing (13.3), 202-205
EditContactView.xaml listing (14.5), 221-226
EditContactView.xaml.cs listing (13.5), 205-206
EditingCommands built-in command library, 146
Editor Tabs (ZAM 3D), 428
elements, trees, 121
ellipses, drawing, 299
EmbossBitmapEffect, 342
ending animations, 393
entities, building, 182-183
equations, easing, timelines, 387-388
Evans, Eric, 191
event handlers, 68
adding, 124
events, 142
bubbling up, 122
handling, Text Editor, 128-138
naming conventions, 234
preview events, 139-141

routed events, 121-122
RoutedEventArgs, 122-127
using, 127-128
tunneling down, 122
EventTrigger, 377, 379
Exact Positioning with Canvas listing (4.3), 53-54
Expander, 181
Express Edition, Visual Studio 2008, 13
Expression Blend, 13, 424
Expression Design, 14
Extrusion Editor (ZAM 3D), 431

F

factoring
resource files, 216
styles, 226-227
fancyButtonStyle listing (22.1), 377-379
FCL (Framework Class Library), 12
Figure inline element (TextBlock), 61
file formats, applications, 93
files
code-behind files, 29-30
resource files, factoring, 216
standard executable files, 93
XAML files, 94
converting applications to, 94-95
renaming, 30-32
XBAP files, 93
converting applications to, 96-97
publishing, 97-98
FillBehavior property, 392
fills
animations, 393
drawing, 295-297
finalizing media player, Media Viewer, 286-288
Flash, 425
Floater inline element (TextBlock), 61
FlowDocumentPageViewer document control, 168
FlowDocumentPageViewer named part (controls), 358
FlowDocumentReader document control, 168
FlowDocumentReader named part (controls), 358
FlowDocumentScrollViewer document control, 168
FlowDocumentScrollViewer named part (controls), 359
FocusManager.FocusedElement, 64
font collections, binding to, 80
Font Viewer
building, 27-39
code, downloading, 437
fancy instructions, 61
font collections, binding to, 80
laying out, 41-57
user controls, 100-104
XAML file, converting to, 94-95
XAML-only version, 402
XBAPs, converting to, 96-97
Font Viewer with a User Control listing (7.2), 103
formatted text support, text document editors, adding to, 115-118
formatting bound data, 239-247
forms, contact forms, 68, 70
fps (frames per second), 383
Frame named part (controls), 359
frame rate, animation, 383
frames, animation, 383
frame rate, 383
key frames, 384, 394-396
tweens, 384
Framework Class Library (FCL), 12
FrameworkElement base class, 42

functions. *See also* controls
- Alignment, 42-44
- Button, 67-68
- Canvas, 53-54
- Decorators, 54-56
- DockPanel, 46-48
- Grid, 48-52
- Labels, 64-67
- ListBox, 71
- Margin, 42-44
- Padding, 44
- Panels, 45
- StackPanel, 45-47
- TextBlock, 60-64
- TextBox, 61-64
- ToggleButtons, 68-71
- WrapPanel, 52-54

G

Gallery Toolbar (ZAM 3D), 428
gathering text from users, 61-64
geometric transfers
- BitmapEffect, 339-341
  - adding to Media Viewer, 342-344
  - BevelBitmapEffect, 342
  - BitmapEffectGroup, 342
  - BlurBitmapEffect, 342
  - DropShadowBitmap-Effect, 341
  - EmbossBitmapEffect, 342
  - OuterGlowBitmapEffect, 341
- Media Viewer, adding to, 337-339

geometric transforms, 331, 333
- LayoutTransform, 335-337
- RenderTransform, 335-337
- RotateTransform, 334
- ScaleTransform, 334
- SkewTransform, 333
- TransformGroup, 334
- TranslateTransform, 333

gestures, 145
- commands, defining, 155

graphical user interfaces (GUIs), 5
graphics, 7-8
Grid function, 48-52
GridSplitter, 52
GridViewColumnHeader named part (controls), 359
GUIs (graphical user interfaces), 5

H

Handled property (RoutedEventArgs), 123
handling events, Text Editor, 128-138
headers, Media Viewer, creating, 299-302
“hello world” XAML applications, creating, 18
hierarchies, classes, 171
Hierarchy Toolbar (ZAM 3D), 427
Hyperlink inline element (TextBlock), 61

I

icons, 426
- Media Viewer, replacing, 437

Implementing the DocumentManager listing (8.3), 115-117
Ingebretsen, Robby, 423
inheritors, TriggerAction, 379
Inline elements (TextBlock), 61
InlineUIContainer inline element (TextBlock), 61
INotifyPropertyChanged, 234
input bindings, commands, creating, 153-155
instances, rendering, 253-254
IntelliSense, 33
- event handlers, adding, 124

interfaces
- decorators, enhancing with, 54-56
- GUIs (graphical user interfaces), 5
- ZAM 3D, 427-428

interpolation, key frames, 395

inversion of control, application design, 410-411

Italic inline element (TextBlock), 61

ItemContainerGenerator class, 254

items, displaying, 253

items controls, panels, choosing in, 351-352

ItemsControl, 201, 252-256, 262

- instances, rendering, 253-254
- items, displaying, 253
- lists, laying out, 256
- Selector class, 260-261
- side bars, customizing, 256, 258-259

ItemsControl with ItemTemplate listing (16.1), 256-257

J–K

Kaxaml, 14, 423

key frames

- animation, 384
  - interpolation, 395
- animations, 394-396

keyed styles, 220-221, 226

Kuler utility, 317

L

Label, 64-67

- TextBlock, compared, 67

labels, controls, accessing, 64-67

Lathe Editor (ZAM 3D), 429

laying out lists, 256

layout, 57

- Alignment function, 42-44
- applications, 41
  - decorators, 41-42
  - panels, 41-42
  - System.Windows. FrameworkElement base class, 42
- building up, 56
- Canvas function, 53-54
- Decorators function, 54-56
- DockPanel function, 46-48
- Grid function, 48-52
- Margin function, 42-44
- Panels function, 45
- StackPanel function, 45-47
- UIs, 414-415
- WrapPanel function, 52-54

layout (application), creating, 108-109

LayoutTransform, 335-337

leveraging triggers, 369-373

libraries

- BCL (Base Class Library), 12
- built-in command libraries, 146
- FCL (Framework Class Library), 12
- WPF (Windows Presentation Foundation), 6

Lightwave, 424

linear interpolation, key frames, 395

LinearGradientBrush, 319-320

LineBreak inline element (TextBlock), 61

lines, drawing, 292-295

Linq, 242

List View, Contact Manager, displaying, 236-237

ListBox, 71

listings

- 2.1 (Simple Button in XAML), 19-20
- 3.1 (MainWindow.xaml), 37-38
- 4.1 (Simple Grid Layout), 48-49
- 4.2 (Advanced Grid Layout), 50-51
- 4.3 (Exact Positioning with Canvas), 53-54
- 5.1 (Simple Contact Form XAML), 62-63
- 5.2 (More Navigable Contact Form), 64-66
- 5.3 (More Usable Contact Form), 68-70
- 6.1 (Binding a Slider to a TextBlock), 77
- 6.2 (Adding a ToolTip to the Font Viewer), 83-84

7.1 (Markup for FontViewer.xaml), 95-96
7.2 (Font Viewer with a User Control), 103
8.1 (Toolbars in a UserControl), 112
8.2 (Menu Contained by a UserControl), 114
8.3 (Implementing the DocumentManager), 115-117
9.1 (Markup for MainWindow.xaml in the Text Editor), 132-136
9.2 (Code-Behind for TextEditorToolbar.xaml), 136-138
9.3 (Code-Behind for MainWindow.xaml), 138
9.4 (Window1.xaml Captures All the KeyDown Events), 140-141
9.5 (Window1.xaml.cs Implements a General Handler), 141
10.1 (Markup for TextEditorToolbar.xaml), 147-148
11.1 (Custom DocumentPaginator), 158-160
11.2 (PrintManager), 161-163
11.3 (PrintPreviewDialog.xaml), 165
11.4 (PrintPreviewDialog.xaml.cs), 166-167
11.5 (App.xaml.cs), 169-170
12.1 (Shell.xaml), 180-181
12.2 (Address.cs), 183-187
12.3 (Contact.cs), 184-187
12.4 (States.cs), 188
12.5 (ContactRepository.cs), 188-190
13.1 (PresenterBase.cs), 194
13.2 (ApplicationPresenter.cs), 197-199
13.3 (EditContactView.xaml), 202-205
13.4 (EditContactPresenter.cs), 204-206
13.5 (EditContactView.xaml.cs), 205-206
14.1 (App.xaml), 212
14.2 (SearchBar.xaml), 214
14.3 (SearchBar.xaml.cs), 214-215
14.4 (DefaultStyles.xaml), 217-218
14.5 (EditContactView.xaml), 221-226
15.1 (Converter in Application Resources), 243
15.2 (PhoneConverter Class), 245-246
16.1 (ItemsControl with ItemTemplate), 256-257
17.1 (ApplicationController.cs), 269-270
17.2 (MainWindow.xaml.cs), 270
17.3 (MenuView.xaml), 272
17.4 (MenuPresenter.cs), 272-273
17.5 (MenuView.xaml.cs), 273-274
17.6 (Media.cs), 274-276
17.7 (Picture.cs), 275-276
17.8 (PictureView.xaml), 279-280
17.9 (MediaPresenter.cs), 280-281
17.10 (MediaPlayer.xaml), 282-284
17.11 (MediaPlayer.xaml.cs), 283
17.12 (MusicView.xaml), 286-287
17.13 (VideoView.xaml), 288
18.1 (Several Lines in Window1.xaml), 294-295
18.2 (Polyline, Polygons, and Fills), 297-298
18.3 (MainWindow.xaml with Some Styling), 302-303
20.1 (Window1.xaml), 332-333
20.2 (Window1.xaml with LayoutTransform), 335-336
20.3 (Window1.xaml with BitmapEffect), 339-340

21.1 (Window1.xaml), 363
21.2 (Window1.xaml Refactored), 364-365
22.1 (fancyButtonStyle), 377-379
23.1 (Window1.xaml), 385
23.2 (Window1.xaml Demonstrating Animation with a Path), 397
23.3 (Window1.xaml Demonstrating a Controlled Animation), 399

lists
classes, 252
control hierarchy, 251-252
displaying, 71
ItemsControl, 252-256
laying out, 256
side bars, customizing, 256-259

local animations, 400

M

MainWindow.xaml, 37-38, 270
markup for, 132-136
styling, 302-303

MainWindow.xaml listing (3.1), 37-38

MainWindow.xaml with Some Styling listing (18.3), 302-303

MainWindow.xaml.cs listing (17.2), 270

manual animations, 401

Margin function, 42-44

markup
MainWindow.xaml, 132-136
UIs, 226

markup extensions (XAML), 22-23

Markup for FontViewer.xaml listing (7.1), 95-96

Markup for MainWindow.xaml in the Text Editor listing (9.1), 132-136

Markup for TextEditorToolbar.xaml listing (10.1), 147-148

markup language for declarative UI, 5

Maya, 425

media controls, Media Viewer, styling, 303-306

media player, Media Viewer
creating, 282-285
finalizing, 286-288

Media Viewer, 267, 289
animations, 388-390
acceleration, 392-393
applying to, 401-405
clock animations, 400
controlling, 398-400
deceleration, 392-393
fills, 393
key frames, 394-396
local animations, 400
manual animations, 401
path geometry, 396-398
placement, 391-392
properties, 392-394
repeating, 393
target property resolution, 390-391
bitmap effects, adding to, 342-344
brushes, 318-319
DrawingBrush, 323-325
LinearGradientBrush, 319-320
RadialGradientBrush, 321-323
SolidColorBrush, 319
VisualBrush, 326-327
code, downloading, 437
colors, 315-316
choosing, 316-318
control templates
adding to, 353-358
benefits, 366
creating, 362-366
creating for a slider, 360-362
identifying special parts, 358-362
fills, drawing, 295-297
geometric transforms, 331, 333
adding to, 337-339
bitmap effects, 339-342
LayoutTransform, 335-337

RenderTransform, 335-337
RotateTransform, 334
ScaleTransform, 334
SkewTransform, 333
TransformGroup, 334
TranslateTransform, 333
header, creating, 299-302
icons, replacing, 437
media controls, styling, 303-306
media player
creating, 282-285
finalizing, 286-288
menu screen, implementing, 271-274
model, building, 274-277
paths geometry, 306-310
picture screen, creating, 277-281
requirements, defining, 267
shapes, 291
ellipses, 299
lines, 292-295
polygons, 296-298
polylines, 295-298
rectangles, 299
strokes, 292-295
solution, setting up, 268-271
stream geometry, 310-311
triggers
advanced scenarios, 380-381
DataTrigger, 373-377
EventTrigger, 377-379
leveraging, 369-373
restrictions, 380
Media.cs listing (17.6), 274-276
MediaCommands built-in command library, 146
MediaPlayer.xaml, 282-285
MediaPlayer.xaml listing (17.10), 282-284
MediaPlayer.xaml.cs listing (17.11), 283
MediaPresenter.cs listing (17.9), 280-281
Menu Contained by a UserControl listing (8.2), 114
menu screen, implementing, 271-274
MenuItem named part (controls), 359
MenuPresenter.cs listing (17.4), 272-273
menus
commands, binding to, 148-149
text document editors, adding to, 113-115
MenuView.xaml listing (17.3), 272
MenuView.xaml.cs listing (17.5), 273-274
MessageBox class, 71
methods
CloseTab, 199
ConvertBack, 240, 244-245
Search, 199
Microsoft Visual C# 2008 Express Edition, 28
Mobiform Software, Aurora, 14, 424
modal dialog boxes, 71
Model-View-Controller (MVC), 178
Model-View-Presenter pattern, 178-179
models
defining, 182-187
Media Viewer, building, 274-277
modes, data binding, 79
Mole, 423
More Navigable Contact Form listing (5.2), 64-66
More Usable Contact Form listing (5.3), 68-70
motion, 383
MusicView.xaml listing (17.12), 286-287
MVC (Model-View-Controller), 178
MVP (Model View Presenter), application design, 413
MVP pattern, 179, 193

N

Name property (Button class), 20
named parts, controls, 358-360
namespaces, XML, 19
naming conventions, events, 234
navigation applications, 99-100
navigation models, applications, 98-100
NavigationCommands, 146
NavigationWindow named part (controls), 359
.NET Framework, 1, 5, 12
  version history, 13
new projects, setting up, 28-29
notifications, automatic change notifications, demonstrating, 85-88

O

observable collections, demonstrating, 235-236
OneTime mode (data binding), 79
OneWay mode (data binding), 79
OneWayToSource mode (data binding), 79
OriginalSource property (RoutedEventArgs), 123
orthogonal code, applications, designing, 408, 411
OuterGlowBitmapEffect, 341
output, printing
  custom document paginator, 158-160
  enabling, 157-158
  Print Manager, 161-164
  Print Preview, 164-168
outputting bitmaps, 168-172

P

Padding functions, 44
panels, items controls, choosing in, 351-352
panels (layout), 41-42
Panels function, 45
parameterizing converters, 246-247
partial classes, 30
PasswordBox named part (controls), 359
path geometry, animations, 396-398
patterns
  application design, 409-412
  Model-View-Presenter pattern, 178-179
  MVP pattern, 179, 193
PauseStoryboard inheritor (TriggerAction), 379
Penner, Robert, 388
pens, darwing with, 307
PhoneConverter Class listing (15.2), 245-246
picture screen, Media Viewer, creating, 277-281
Picture.cs listing (17.7), 275-276
PictureView.xaml listing (17.8), 279-280
placing animations, 391-392
polygons, drawing, 296-298
Polyline, Polygons, and Fills listing (18.2), 297-298
polylines, drawing, 295-298
PresenterBase.cs class, creating, 193-194
PresenterBase.cs listing (13.1), 194
preview events, 139-141
primary colors, 317
Print Manager, developing, 161-164
Print Preview, applications, adding to, 164-168
printing
  custom document paginators, implementing, 158-160
  Text Editor, enabling, 157-158
PrintManager listing (11.2), 161-163
PrintPreviewDialog.xaml listing (11.3), 165

PrintPreviewDialog.xaml.cs listing (11.4), 166-167
Prism, 425
programming languages
  C#, 1
  XAML, 17-18
    syntax, 19-23
ProgressBar named part (controls), 359
project files (Visual Studio 2008), 29
projects, setting up, 28-29
properties
  animations, 392-394
  BasedOn, 227
  Content property, 21-22
  DataContext, 199
  dependency properties, 84-85, 234
    custom dependency properties, 167
  RoutedEventArgs, 123
  setting, 20-21
  target properties, resolving, 390-391
  TargetType, 218
property elements, 21
property paths, 194
Property Toolbar (ZAM 3D), 428
publishing XBAPs, 97-98

Q–R

RadialGradientBrush, 321-323
raster graphics, vector graphics, compared, 8
rectangles, drawing, 299
Refactor menu commands, Rename, 31-32
refactoring code, 32
RemoveStoryboard inheritor (TriggerAction), 379
Rename command (Refactor menu), 31-32
renaming XAML files, 30-32
rendering instances, 253-254
RenderTransform, 335-337
RepeatBehavior property, 392
RepeatButton (Slider), 360
repeating animations, 393
repository classes, 188
requirements, applications, defining, 267
resolving target properties, 390-391
resource files, factoring, 216
ResourceDictionary, 211
resources
  application design, 415
  combining, 213
  defining, 211-212
  UIs, using in, 214-215
restrictions, triggers, 380
ResumeStoryboard inheritor (TriggerAction), 379
Rich Internet Applications (RIAs), 5, 14
rich text, displaying, 60-61
RichTextBox, 115-118
RotateTransform, 334
Rotation Trackball (ZAM 3D), 427
routed events, 121-122
  RoutedEventArgs, 122-127
  using, 127-128
RoutedEvent property (RoutedEventArgs), 123
RoutedEventArgs, 122-127, 142
Run inline element (TextBlock), 61

S

SaveContact command, 179
scalable graphics, 7-8
ScaleTransform, 334
ScrollBar named part (controls), 359
ScrollViewer control, 201
ScrollViewer named part (controls), 359
Search command, 179
Search method, 199

SearchBar, 215
  Contact Manager, adding to, 214-215
SearchBar.xaml listing (14.2), 214
SearchBar.xaml.cs listing (14.3), 214-215
secondary colors, 317
SeekStoryboard inheritor (TriggerAction), 379
Selection Range (Slider), 361
Selector class, 260-261
separation of concerns (SoC), 199
Separation of Concerns principle, 178
SetStoryboardSpeedRatio inheritor (TriggerAction), 379
Setter, 381
setting properties, 20-21
Several Lines in Window1.xaml listing (18.1), 294-295
shapes, 291
  ellipses, drawing, 299
  headers, drawing, 299-302
  lines, drawing, 292-295
  media controls, styling, 303-306
  path geometry, 306-310
  polygons, drawing, 296-298
  polylines, drawing, 295, 297-298
  rectangles, drawing, 299
  stream geometry, 310-311
  strokes, drawing, 292-295
SharpDevelop, 13
Shell.xaml listing (12.1), 180-181
shells
  application shells, creating, 179-182
  enhancing, 200-201
  solution shells, creating, 179-182
Shifflett, Karl, 423
SideBar.xaml.cs, 200
signatures, 68
Silverlight, WPF, compared, 14-15
Simple Button in XAML listing (2.1), 19-20
Simple Contact Form XAML listing (5.1), 62-63
Simple Grid Layout listing (4.1), 48-49
SkewTransform, 333
SkipStoryboardTo Fill inheritor (TriggerAction), 379
Slider, 182
  functional parts, 360
Slider named part (controls), 359
sliders, templates, creating for, 360-362
Smith, Andrew, 423
Smith, Josh, 423
SoC (separation of concerns), 199
SoC (Separation of Concerns), applications, designing, 408, 412
SolidColorBrush, 319
solution shell, creating, 179-182
solutions, applications, setting up, 268-271
SoundPlayerAction inheritor (TriggerAction), 379
Source property (RoutedEventArgs), 123
Span inline element (TextBlock), 61
SpeedRatio property, 392
splined interpolation, key frames, 395
split pane, Visual Studio 2008, 29
SRP (Single Responsibility Principal), applications, designing, 408-409, 412
StackPanel function, 45-47
standard executable files, 93
States.cs listing (12.4), 188
StaticResource markup extension, 23
  DynamicResource markup extension, compared, 216
StickyNoteControl named part (controls), 359

StopStoryboard inheritor (TriggerAction), 379
strokes, drawing, 292-295
Style, 381
style sheets, 5
styles, 227
- application design, 415-416
- defining, 217-220
- factoring, 226-227
- keyed styles, 220-221, 226

StyleSelector class, 255
styling media controls, Media Viewer, 303-306
SVG, 425
syntax
- data binding, 76, 89
- XAML, 19-23

System.Windows.Controls. Control base class, 42
System.Windows.Framework-Element base class, 42

T

tabbed UI infrastructure, completing, 194-196
TabControl, 182
TabControl named part (controls), 359
target properties, resolving, 390-391
TargetType property, 218
TDD (Test Driven Development), 413
templates, 8-9
- application design, 416-417
- control templates, 347-350
  - adding to applications, 353-358
  - benefits, 366
  - controlling, 351-352
  - creating, 362-366
  - creating for a slider, 360-362
  - identifying special parts, 358-362
  - items controls, 351-352
  - placing content in, 350-351
- data templates, 237-239
  - applying, 259
- sliders, creating for, 360-362

templating, 59
tertiary colors, 317
Test Driven Development (TDD), 413
text, 60
- controls, accessing with labels, 66-67
- rich text, displaying, 60-61
- users, gathering from, 61, 63-64

text document editors, 107
- application layout, creating, 108-109
- events, handling, 128-138
- formatted text support, adding, 115, 117-118
- menus, adding, 113-115
- toolbars, adding, 109-110
- user controls, adding, 110-113

Text Editor, 107, 118
- application layout, creating, 108-109
- bitmaps, outputting, 168-172
- code, downloading, 437
- commands, adding to, 151-152
- custom document paginator, implementing, 158, 160
- events, handling, 128-138
- formatted text support, adding, 115-118
- MainWindow.xaml, markup for, 132-136
- menus, adding, 113-115
- Print Manager, developing, 161-164
- Print Preview, adding to, 164-168
- printing, enabling, 157-158

toolbars, 146-147
adding, 109-110
user controls, adding, 110-113

TextBlock
Inline elements, 61
Label, compared, 67
rich text, displaying, 60-61
users, gathering text from, 61-64

TextBox, users, gathering text from, 61-64

Thumb (Slider), 360

TickBar (Slider), 360

timelines, animation, 384-388

ToggleButtons, 68-71

toggling buttons, 68-71

ToolBar named part (controls), 359

toolbars
text document editors, adding to, 109-110
Text Editor, 146-147

Toolbars in a UserControl listing (8.1), 112

tools
application frameworks, 425
designer tools, 424
developer tools, 423
XAML converters, 424-425

tools (WPF), 13-14

ToolTip, 64, 82

Track (Slider), 360

TransformGroup, 334

transforms, 331-333
BitmapEffect, 339-341
BevelBitmapEffect, 342
BitmapEffectGroup, 342
BlurBitmapEffect, 342
DropShadowBitmap-Effect, 341
EmbossBitmapEffect, 342
Media Viewer, 342-344
OuterGlowBitmapEffect, 341
LayoutTransform, 335-337
Media Viewer, adding to, 337-339
RenderTransform, 335-337
RotateTransform, 334
ScaleTransform, 334
SkewTransform, 333
TransformGroup, 334
TranslateTransform, 333

TranslateTransform, 333

trees, elements, 121

TreeViewItem named part (controls), 359

TriggerAction, inheritors, 379

triggering actions, buttons, 67-68

triggers, 9, 369, 381
advanced scenarios, 380-381
DataTrigger, 373-377
EventTrigger, 377-379
leveraging, 369-373
restrictions, 380

tunneling down events, 122

tweens, animation, 384

two controls, binding, 76-77

two-way data binding, 78-85

TwoWay mode (data binding), 79

U

UIs (user interfaces)
application design, 412-414
layout, 414-415
declarative UI, 7
decorators, enhancing with, 54-56
GUIs (graphical user interfaces), 5
markup, 226
resources, using in, 214-215
tabbed UIs, completing, 194-196

Underline inline element (TextBlock), 61

unified APIs, 6

UniformGrid, 341

usability, application design, 414

user controls, 100-104
- custom controls, compared, 101
- text document editors, adding to, 110-113

user needs, anticipating, 131

users, test, gathering from, 61-64

V

value objects, building, 182-183

VB .NET 9.0, 12

vector graphics, 7
- raster graphics, compared, 8

version history, .NET Framework, 13

VideoView.xaml listing (17.13), 288

Viewports (ZAM 3D), 427

Visual Basic applications
- adding formatted text support to, 115-118
- adding menus to, 113-115
- adding user controls to, 110-113

Visual Basic .NET, 12

Visual C# 2008 Express Edition, 28

Visual Studio
- application layout, creating, 108-109
- XAML files, converting applications to, 94
- XBAPs
  - converting applications to, 96-97
  - publishing, 97-98

Visual Studio 2008, 423
- applications, developing, 33-39
- code-behind files, 29-30
- Express Edition, 13
- new projects, setting up, 28-29
- project files, 29
- split pane, 29
- XAML files, renaming, 30-32

VisualBrush, 326-327

W

WCF (Windows Communication Foundation), 6

WF (Workflow Foundation), 6, 17

Window1.xaml Captures All the KeyDown Events listing (9.4), 140-141

Window1.xaml Demonstrating a Controlled Animation listing (23.3), 399

Window1.xaml Demonstrating Animation with a Path listing (23.2), 397

Window1.xaml listing (20.1), 332-333

Window1.xaml listing (21.1), 363

Window1.xaml listing (23.1), 385

Window1.xaml Refactored listing (21.2), 364-365

Window1.xaml with BitmapEffect listing (20.3), 339-340

Window1.xaml with LayoutTransform listing (20.2), 335-336

Window1.xaml.cs Implements a General Handler listing (9.5), 141

Windows Communication Foundation (WCF), 6

Windows Forms, 6

Windows Presentation Foundation (WPF). ***See*** **WPF (Windows Presentation Foundation)**

WinForms, 1

Workflow Foundation (WF), 6, 17

WPF (Windows Presentation Foundation), 1, 5, 15
- 3D animation, 10
- 3D modeling, 10
- animation framework, 10
- ASP.NET, compared, 11

benefits, 10-11
data binding, 9
declarative UI, 7
layout, 7
markup extensions, 23
scalable graphics, 7-8
sibling libraries, 6
Silverlight, compared, 14-15
styling, 9
templates, 8-9
tools, 13-14
triggers, 9
WPF CAB, 425
WrapPanel function, 52-54

X-Z

X:Array markup extension, 23
x:Null markup extension, 23
x:Type markup extension, 23
XAML, 7, 14, 17-18, 24
application design, 418
collection views, 248-249
markup extensions, 22-23
properties
Content property, 21-22
setting, 20-21
syntax, 19-23
XAML converters, 424-425
XAML files, 94
applications, converting to, 94-95
renaming, 30-32
XAML-only Font Viewer, 402
XBAP files, 93
applications, converting to, 96-97
publishing, 97-98
XML namespaces, 19

YAGNI (Ya Aren't Gonna Need It), 131
application design, 411

ZAM 3D, 424, 427
carrot animation, creating, 428-435
Extrusion Editor, 431
interface, 427-428
Lathe Editor, 429