

Expression Blend IN ACTION

Joel Cochran



MEAP

 MANNING



**MEAP Edition
Manning Early Access Program
Expression Blend in Action MEAP version 11**

Copyright 2012 Manning Publications

For more information on this and other Manning titles go to
www.manning.com

Table of Contents

Preface: My Blend Story

Part 1: Getting started with Blend

- 1 Introducing Expression Blend
- 2 Demystifying Blend
- 3 Working with elements
- 4 Container-driven design—understanding layout
- 5 Data binding in Blend

Part 2: Getting jiggy with it: styling applications

- 6 Styles: adding flair to elements
- 7 Templates: crafting control structure
- 8 Resources and resource dictionaries
- 9 Mastering the Visual State Manager

Part 3: Taking Blend to the next level

- 10 Animations and storyboards
- 11 Enhancing your BlendFu with custom behaviors
- 12 Discovering your inner artist

Appendixes

- A A brief introduction to MVVM

1

Introducing Expression Blend

In this chapter we will be

- Learning what Blend is and how to get it
- Seeing how Blend fits into the Development story
- Learning how to use Blend and Visual Studio together
- Taking a high level tour of Blend
- Creating your first project in Blend

Microsoft Expression Blend is the graphic design tool of choice for eXtensible Application Markup Language (XAML) based applications. Unlike previous technologies such as Visual Basic 6 (VB6) and Windows Forms (WinForms), the XAML approach separates User Interface (UI) code and application execution into distinct files. Blend is a visual design tool that generates XAML, increases your productivity, provides rapid prototyping, and greatly simplifies visual tasks.

Microsoft introduced Windows Presentation Foundation (WPF) in 2007. WPF represented the first significant change to Graphical User Interface (GUI) technology since its inception at Xerox Parc almost 25 years before. Previous technologies were essentially bitmap managers, but WPF uses vector graphics to display GUI. Vector graphics are mathematically calculated drawings that can resize almost instantly and infinitely with no pixelation or loss of resolution. This new approach promised to solve many of the problems that plagued traditional windows design associated with screen resolutions, resizing, scaling, and more.

Along with this radically different concept came a different way to create GUI design, through a new XML specification called XAML. At a high level XAML resembles HTML but it doesn't take long to discover that this specification is far more complex and powerful. The huge differences between XAML and traditional methods required a new way to design UI.

To create vector oriented designs, the logical step was to produce a vector oriented design tool. Microsoft Expression Blend is that tool.

In this chapter we are going to take a look at the key players involved in this exciting technology and see how they relate to one another. Then we'll learn what Blend is and where it fits into the mix. We'll take a high level look at the tool itself and then we'll jump right in and create a project using Blend and Visual Studio together.

1.1 The key players

With the advent of XAML based technologies developer can now use the same tools and methods to create both desktop applications and Rich Internet Applications (RIA). While the delivery platforms are different the execution layer of both is now client based. This means, for the first time, if you can develop applications for the desktop you can also effectively develop applications for the web and vice versa.

Blend is used to edit XAML which is the common thread between Windows Presentation Foundation (WPF) and Silverlight. To better understand these relationships, let's take a brief look at WPF, Silverlight, and XAML. We'll even throw in a little Visual Studio for good measure.

1.1.1 WPF

Windows Presentation Foundation (WPF) is the XAML technology geared towards desktop application development. WPF is not an evolution of WinForms: it is a wholly separate technology part of the .NET Framework 3.5 and later. As a first class citizen, WPF executes in full trust and has access to the complete framework. It has networking capabilities, supports direct database access through technologies like ADO.NET and Entity Framework, local and network file IO, and more.

WPF is the right choice for applications that require framework elements not provided by Silverlight, 3D support, full access to the local client machine, network access, and other features traditionally available to desktop applications. WPF is currently only supported on Windows operating systems. While WPF does have a browser execution format called XBAP, even in those scenarios the client machine must run Windows with .NET Framework installed, and the application must run in Internet Explorer. All of these factors should play into your decision to use WPF.

1.1.2 Silverlight

Silverlight is a cross-platform, cross-browser technology used to create Rich Internet Applications (RIA). In this context Silverlight is considered an Adobe Flash competitor. Both install in the browser via a plug-in, and both offer rich support for vector graphics, animations, and media, but that's where the similarities end.

Silverlight's number one asset is the fact that it ships a Common Language Runtime (CLR) that is essentially a miniaturized version of the .NET Framework. This CLR is much smaller than the full framework and optimized to run in the constraints of the Internet

sandbox. This feature allows you to write .NET code, in either C# or Visual Basic .NET, and ship it along with your application. Unlike WebForms, this code executes entirely on the client and as a result it performs very well. A Silverlight application can even be installed to the local machine and execute as a desktop application with elevated trust. It competes with Adobe Air in this context but has more features and more flexibility.

There is a common misconception that Silverlight is a subset of WPF but this is not the case. The Silverlight CLR is a standalone entity and does not have a reliance on the .NET Framework. Users do not have to have the .NET Framework installed, they do not have to use Internet Explorer, nor do they have to be running Windows because Silverlight is supported on Apple Mac OS through a Safari plug-in.

Silverlight for Windows Phone 7

WP7 is nothing like previous Windows Mobile efforts which had largely been panned and abandoned by the consumer market. With a completely new concept, called Hubs, and a UI inspired by the Zune's critically acclaimed Metro interface, WP7 is squarely focused on consumers. In order to successfully compete against the iPhone and Droid, WP7 needed to have great applications with sleek and modern UIs. The technologies Microsoft chose to achieve this were XNA and Silverlight. XNA, the development platform for Xbox, is great for developing games, but most WP7 developers turned to Silverlight to create the next generation of mobile applications. The version of Silverlight for WP7 is a hybrid of Silverlight 3, Silverlight 4, and Phone specific features.

In order to support and encourage WP7 Silverlight development Microsoft produced a new version of Blend, Microsoft Expression Blend for Windows Phone. Blend for Phone includes templates and controls specific to the WP7 form factor, pre-styled for the Metro interface. Most notable of these are the Pivot and Panorama controls which give many applications their distinctive WP7 feel. Blend uses the WP7 emulator, installed by the Windows Phone Developer tools, to test and execute applications locally. You can find information specific to Blend for Phone in Appendix E.

1.1.3 XAML

Both WPF and Silverlight use an XML specification called eXtensible Application Markup Language (XAML) to describe and express their user interfaces. At a high level XAML resembles HTML but it doesn't take long to discover that this specification is quite a bit more complex and powerful. While WinForms and WebForms shared things like .NET and languages, they never shared the GUI layer like WPF and Silverlight do.

This connective tissue allows developers of either technology to easily become proficient in the other. XAML incorporates more than just UI, it also encompasses many User Experience (UX) features. I think of UI as what the user sees and UX as how the application visually reacts to what the user does. Mouseovers, animations, states, transitions,

visualizations, and other UX related functions are all typically described in XAML. XAML melds both UI and UX into a single descriptive language.

XAML is also a key player in one of WPF and Silverlight's chief goals: separation of concerns. True separation of concerns was essentially a pipe dream in previous technologies because the code to describe the user interface and the code to execute it were typically in the same file. Since XAML is its own file type it is by necessity separate from the code that executes. While there is still a code behind file that accompanies it, good XAML together with solid application architecture can almost eliminate the need for code behind.

In order to encourage better architecture and less code behind, XAML provides support for application features such as Data Binding and Commands. With all the stuff going on in XAML you can easily understand that it can get very complex very quickly. Listing 1.1 shows a small chunk of XAML from a trivial application.

Listing 1.1 The XAML for the MouseOver state of a ControlTemplate for a Button.

```
<VisualState x:Name="MouseOver">
  <Storyboard>
    <ColorAnimationUsingKeyFrames Storyboard.TargetProperty="(TextElement[CA]nt.Foreground).(GradientBrush.GradientStops)[1].(GradientStop.Color)"
      [CA]Storyboard.TargetName="textBlock">
      <EasingColorKeyFrame KeyTime="0"
        Value="#69FFFFFF" />
    </ColorAnimationUsingKeyFrames>
    <ColorAnimationUsingKeyFrames Storyboard.TargetProperty="(TextElement[CA]nt.Foreground).(GradientBrush.GradientStops)[0].(GradientStop.Color)"
      [CA]Storyboard.TargetName="textBlock">
      <EasingColorKeyFrame KeyTime="0"
        Value="#FFDAFF00" />
    </ColorAnimationUsingKeyFrames>
  </Storyboard>
</VisualState>
```

This is a small section of a template for a Button in the application. The full template is about 100 lines long. The entire 322 line listing, which contains three simple graphics (whose code is particularly ugly), one Button, and a short animation, is available in the code samples for the book.

If the code above seems confusing and tedious to you don't worry, I agree. This is the real power of Blend: it took me about 10 minutes to throw the entire application together in Blend and I didn't write a single line of XAML. I can't imagine how long it would take me to code this in Visual Studio.

1.1.4 Visual Studio

Regardless of how much I use and love Blend, at the end of the day I am still a developer. The bulk of my time is spent in application development and maintenance, work that is naturally done in Visual Studio. That being said I have no desire to hand code GUI, XAML or otherwise, in a text editor.

While Visual Studio is completely capable of editing XAML I find it a laborious, slow, tedious, and frustrating task. There are many aspects of XAML I would not feel comfortable writing in a text editor, like animation storyboards and complex gradients. Fortunately, I don't have to thanks to Expression Blend.

1.2 Enter Expression Blend

In listing 1.1 above we got a quick glimpse of what XAML looks like. This sample is fairly trivial and really includes nothing complex, but it immediately leads me to the following question: who would want to code that by hand?

Many times I get presented with the argument that one can create XAML in Visual Studio and so Blend is unnecessary, especially since it isn't free. I'm happy to accept the premise of that argument as long as the developer presenting it will agree that he doesn't need Visual Studio. Every computer and every platform has text editors included at no additional cost, so why don't we write modern applications in Notepad? For users who have learned Blend that's exactly what writing XAML in Visual Studio feels like. Even with the myriad of Visual Studio improvements made to improve XAML design work it still pales in comparison to Blend.

There is a reason that Blend is considered a Design tool: good GUI design is art. Imagine Leonardo da Vinci painting the Mona Lisa. He has a vision of the colors, the scale, the scope, even the individual brush strokes required to see his vision come to life. Now imagine he had to describe that vision in text instead of actually drawing it: do you think he ever would have created it? How about the Sistine Chapel?

NOTE Blend is a Design tool, but that does not mean it is only for designers! As a developer I find Blend indispensable in my work.

So the comparison is a little contrived, but to an experienced Blend user that's what creating complex UI feels like in XAML and Visual Studio. Do you really want to spend your time typing XAML or would you rather use state of the art graphical software? It really is a case of using the right tool for the job: we should use a graphical tool for a graphical task. Doing so not only makes more sense, you can produce these graphical elements much faster making you more productive.

In this section you'll learn how to get Blend and why Blend is the world's best XAML editor. We'll talk about how Blend can increase your productivity and provide rapid prototyping support. Then we'll finish with a discussion on using Blend and Visual Studio together including when to use Visual Studio and when to use Blend. We've got a lot to cover, so let's get started!

1.2.1 How to get Blend

Blend is part of the Microsoft Expression Studio, a suite of applications covering a wide variety of interactivity, prototyping, and application design features. The roster of

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=730>

applications in this collection has changed a few times over the years. The current incarnation includes Blend + SketchFlow, Design, Web + SuperPreview, and Encoder Pro.

One change that occurred in version 3 of Expression Studio was the pricing model. Prior to this release you could purchase the tools separately. Unfortunately this is no longer the case: you can currently only acquire Blend as part of Expression Studio Ultimate. The full retail price of the Studio is \$599, but there are several ways you can acquire it without paying full retail.

Expression Studio Ultimate is included with a Microsoft Developer Network (MSDN) Ultimate subscription. MSDN Premium includes it as well but with one important distinction: the version of Blend included with Premium does not include SketchFlow. If you have a Premium MSDN subscription and wish to upgrade Expression Studio to include SketchFlow without upgrading your MSDN Subscription you can purchase the upgrade through MSDN for \$199.

If you do not have MSDN, you can purchase Expression Studio directly through the Expression website (<http://expression.microsoft.com/en-us/default.aspx>). If you are going to purchase Expression Studio directly, be sure to investigate upgrade pricing because many products, like Visual Studio and some Adobe products, give you access to the upgrade price of \$349. This is a significant savings. Free trial downloads of all the products are available and good for 60 days without a license.

The Expression Studio

Here are brief summaries of the rest of the products in the Expression Studio.

Expression Design

Expression Design is a full featured Vector Graphics design tool, which we'll cover in more detail in Appendix B. As a Blend user there are several reasons you might find Design interesting. Blend and Design share many characteristics making it fairly painless for Blend users to learn Design. Design has a few graphic features that Blend doesn't that occasionally come in handy, such as Layers. Any art created in Design can be exported to XAML and consumed in Blend. Using Layers you can optionally export multiple items into a single Resource Dictionary. While you may not use it frequently there are occasionally scenarios where Design is big help.

Expression Web

Nothing like its FrontPage ancestor, Expression Web is a general, standards based web site design tool that supports multiple languages. In addition to standard web languages like XHTML, CSS, and JavaScript, Web includes support for ASP.NET, ASP.NET AJAX, and even PHP. Web 4 includes an integrated Search Engine Optimization (SEO) reporting tool to help maximize your search engine placement. Perhaps the most notable feature of

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=730>

Web is SuperPreview, which allows you to preview your website in multiple browsers for compatibility testing. IE6, IE7, IE8, FireFox 3.6, and Macintosh Safari are all supported.

Expression Encoder Pro

Encoder Pro is a full featured video editing and encoding tool that supports both VC-1 and H.264 codecs. Included are all the standard editing features such as trimming, clipping, and cutting video, splicing video segments, and inserting graphical overlays. Support for multiple caption file formats is included. A variety of predefined Silverlight output templates are available also, allowing you to create a highly stylized Silverlight page to host your video. And speaking of Silverlight, encoding settings are also available to take advantage of Silverlight Smooth Streaming. Finally, an SDK is included for creating custom encoding tasks.

1.2.2 The world's best XAML editor

Blend is the world's best XAML editor because it relieves me from the task of *actually editing* the XAML. Blend frees me from spending so much time typing XAML syntax and instead allows me to focus on the design work at hand. In my mind it makes perfect sense to use a graphical tool for this task, lest we forget that the "G" in GUI means Graphical! Really, when you think about it the idea of drawing GUI in text makes little sense. The resulting code may be text, but we shouldn't have to type it by hand.

Blend is a vector graphic design tool with a unique function: it translates the drawings on the screen to the expressive text of XAML. I don't want you to get uncomfortable with my use of the word drawing: everything you see on the screen, whether text, a Button, or a logo graphic, is a drawing. While you may think there is more to a Button, the truth is the part that you see is just a drawing. XAML brings true separation of concerns to visual application development. We'll cover this in detail in part 2 when we learn that you can make anything look like anything. This is the magic of Blend and ultimately what makes it a must have tool. Blend is good because of what it does but great because of how well it does it.

Previous attempts at similar tasks include examples like FrontPage. FrontPage was notorious for taking decent looking visual designs and translating them into substandard HTML. The biggest complaint was the insertion of unnecessary code, code that was sometimes indecipherable and extremely difficult to edit successfully outside FrontPage. One of the more frequent questions I receive in my travels is about the quality of the XAML Blend produces. In my opinion it has always been exceptionally lean and clean and only gets better with each release. And remember that XAML is just text, so you can edit it in Visual Studio, 3rd party tools, or even Notepad: I'm just not sure why you'd want to! I'd rather use something like what you see in figure 1.1.



Figure 1.1 Blend showing the application mentioned in section 1.1.3.

I was recently asked if Blend would be considered WYSIWYG. My qualified answer is yes and no. In the classic sense, it is WYSIWYG because what you see in Blend is what you see when the code executes. On the other hand, what you see is the pretty design but what you get is XAML. Also, there are many things you get that you do not see, like Data Binding, so really What You Get Is More Than What You See (WYGIMTWYS), I'm just not sure how you would pronounce that. For our purposes we'll go ahead and consider it WYSIWYG, just understand the topic is open for debate.

1.2.3 Increased productivity

GUI design is considerably faster in Blend thanks to its visual design tooling. Of course, productivity is not entirely about speed, it's about getting more out of the time you spend. Complex XAML tasks such as animations are low hanging fruit, but less obvious items such as defining gradient brushes and other reusable resources are also significantly easier in Blend. Some XAML syntax can be difficult and frustrating: data binding, template definitions, object declarations, and more can all be done in Blend with no editing. This relieves you from fighting syntax and means your time is more productive. Also, these resources can be stored and exported for future reuse. Blend's visualization of resources makes it much easier to organize and reuse these elements.

1.2.4 Rapid prototyping

Rapid prototyping is taken to a whole new level with Blend. Application layout is extremely fast and it is very easy to experiment with different setups and layouts. In WinForms changing a container type, inserting a new container, or restructuring an existing layout were difficult and laborious tasks. With Blend these are very easy so you can feel free to make changes at any point in the process.

Blend creates a standalone GUI, meaning that you can draft a design and execute the application with little or no supporting code. Blend offers several features geared specifically to this level of prototyping. Sample Data is a feature that will create a fake data source, complete with fake values, so that at both design time and during design testing you can see what your data will look like in a real context.

SketchFlow is an application prototyping tool that introduces its own mapping system used to map application flow and navigation. It includes a set of Sketch controls that represent real TextBoxes, ListBoxes, Buttons, etc., but have a distinct hand drawn style. The intent is to focus more on the application features, flow, and layout than the specific look and feel. SketchFlow applications can be hosted and shared over the web using the SketchFlow Player, allowing stakeholders to review the application and provide feedback. This feedback comes in the form of Annotations, a feature that allows notes to be created and shared directly on the Blend artboard. Check out Appendix C for more information about SketchFlow.

Blend is a fantastic tool for design work but we developers spend most of our time in Visual Studio. Let's take a look at how these two tools work together to form a larger experience.

1.3 Using Visual Studio and Blend together

For the designers that work with Blend I expect that integrating with Visual Studio is not a big issue, but for developers adopting Blend it is of primary concern. If you were to see my screen at any given time you would have more than an 80% chance of seeing both Visual Studio and Blend open and simultaneously loaded with the same solution.

Understanding this is key: Blend has no special solutions, projects, or files. The WPF, Silverlight, or WP7 solutions you work on in Blend are the exact same solutions you work on in Visual Studio. While you could theoretically segregate your work into neat little independent tasks requiring only one of the two, I have found that in building real world applications I inevitably need access to both tools at the same time.

Being able to work simultaneously on both design and development tasks is very useful to developers. It allows us to make changes on the fly in code to support the UI, like updating a binding source, or change a UI to reflect a newly coded feature. If Visual Studio is already open with the Solution loaded it makes Blend callbacks, like navigating to Event handlers, much faster.

TIP I recommend using multiple monitors. Having Visual Studio on one monitor and Blend on the other increases the effectiveness of using both tools together.

The issue that naturally arises from such a scenario is synchronization. If I am working on one set of files in two tools I run the risk of losing work by overwriting my progress in one application with the second. To address this risk Microsoft has included features in both tools to prevent any loss of work. Relevant changes in either tool trigger a user prompt to keep the files in synch. We'll see this in action later in this chapter.

1.3.1 When to use Visual Studio

Just like I don't want to write UI in Visual Studio, I don't want to write code in Blend: that's why we have Visual Studio. So even though Blend supports editing XAML, except for the most trivial examples I choose Visual Studio for this task. You may wonder when I would need to edit XAML directly, and while it is increasingly rare, there are still a few XAML tasks that Blend does not support. I also find the occasional case where editing XAML is easier, like setting Grid row heights and column widths to specific values. When I find sample XAML code online it is quicker to simply paste it in to the XAML directly.

NOTE: Just because I use Blend does not mean I throw Visual Studio out the door! In order to be successful you will be using Visual Studio and Blend together.

Blend also supports editing code, meaning you can write C# and VB code in Blend. Ostensibly this is so designers with a little coding skill could edit event handlers or create Behaviors. This has always rubbed me the wrong way and I refuse to use it. Why would I code in Blend when I have the world's best Integrated Development Environment (IDE) at my finger tips? So anytime I need to write code in support of the UI I do so in Visual Studio. Beyond traditional code behind, UI tasks that fall in this category would include Value Converters (chapter 5), Behaviors (chapter 11), and of course architectural elements, like View Models in MVVM (Appendix A).

Blend is capable of creating solutions, adding projects, adding references, adding files, and most of the basic Project management tasks we typically associate with Visual Studio. Over time I have become more agnostic, or more accurately lazy, about which tool I use for these sorts of tasks: in other words I use whichever tool I already happen to be in at the time. When I first started, however, I found it very comforting to limit these tasks to Visual Studio, so you may find this a helpful approach as well.

There is one other scenario I almost always choose Visual Studio for: debugging. Blend does not offer any of the debugging experience we are accustomed to in Visual Studio. While you can execute from either tool I typically execute from Visual Studio. This way if an error occurs or an exception is thrown, I'm already in the right place to handle that situation. Otherwise, if I execute from Blend and an exception is thrown, it means I have to stop the

running process, switch to Visual Studio, and run it again. I find just starting from Visual Studio makes more sense.

1.3.2 When to use Blend

I choose Blend for all UI activities. Adding and managing controls (chapter 3) and creating visual layout (chapter 4) are the most obvious of these tasks. Crafting the look and feel of controls through Styles (chapter 6) and Templates (chapter 7) falls in this category as well.

I also turn to Blend for User Experience (UX) tasks. Application interactivity tasks like Data Binding (chapter 5), Visual States (chapter 9), Behaviors (chapter 10), and Animations (chapter 11), are all faster and easier in Blend. And of course, when I do any graphic design work (chapter 12) then Blend is a no brainer.

1.4 Blend at a glance

In chapter 2 we'll have a thorough walk through of Blend, but for now let's take a look at it from the 10,000 foot level. As we do, you'll see plenty of features Blend and Visual Studio have in common. In this section we'll take a look at the Toolbar and some of the main panels, like Project, Objects and Timeline, and Properties.

1.4.1 The Toolbar

Docked completely on the left hand side of the application, the Blend Toolbar is the most unfamiliar feature to Visual Studio users. In its role as a design tool, and in an effort to appeal to professional designers, Blend's Toolbar was inspired by the toolbars in Adobe products such as Illustrator and Photoshop. Unlike the Toolbox in Visual Studio, Blend's Toolbar does not contain the entire collection of available controls. Instead, it is meant as a quick reference for commonly used drawing tools and controls.

The easiest way to understand the Toolbar is from top to bottom. The upper section of the Toolbar is concerned with managing controls and Blend itself. The center of the Toolbar is oriented towards graphics, drawing, and shapes. The bottom section of the Toolbar is all about application layout and controls. Figure 1.2 shows the Toolbar and highlights these sections.

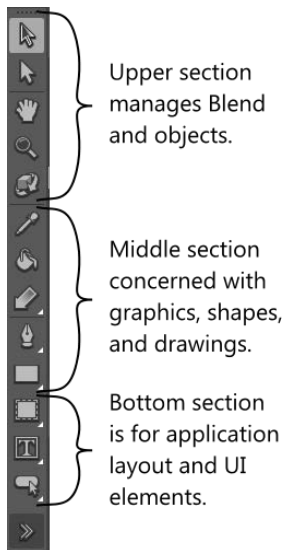


Figure 1.2 The Blend Toolbar as seen in WPF. From top to bottom: managing objects, drawing and graphics, and application layout.

1.4.2 The Projects panel

The Projects panel is found in the upper left hand corner of Blend. It's in a tab panel with some other tabs, like *Assets*, *States*, and *Parts*, so you will need to select it to see its contents. Since both Visual Studio and Blend share the same abilities to manage Solutions and Projects it should come as no surprise that the Projects panel in Blend is very similar to Visual Studio's. In figure 1.3 you can see a Solution with several projects, references, files, and other elements that should make Visual Studio users feel right at home.

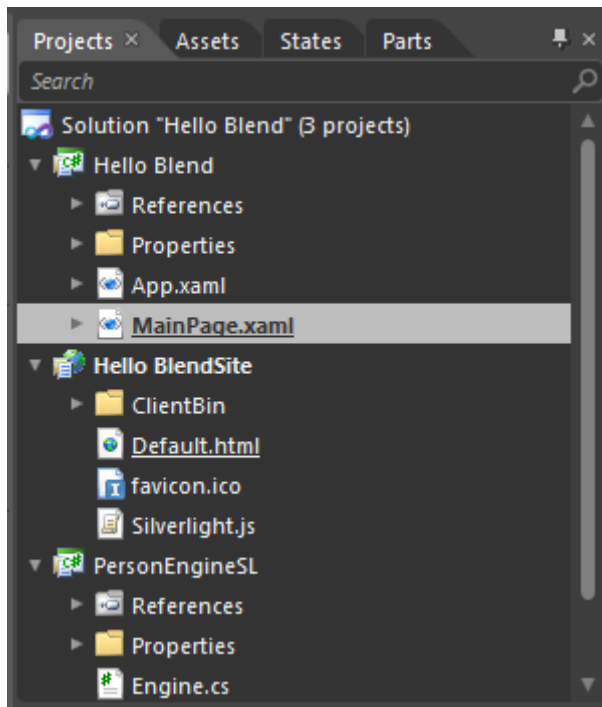


Figure 1.3 The Projects panel in Blend, shown here with 3 projects in the Solution, is very similar to Visual Studio's.

1.4.3 The Objects and Timeline panel

Below the Projects panel is the Objects and Timeline panel. This panel provides a visualization of the object hierarchy. As XAML files become more complex they quickly develop a deep hierarchy of nested containers and elements. This is one of the strengths of XAML but can become difficult to navigate. I often say the Objects and Timeline panel is a developer's best friend because it simplifies navigation and makes it much easier to select the exact element you need. In Chapter 2 we'll see how to make the most of this very important feature, shown in Figure 1.4.

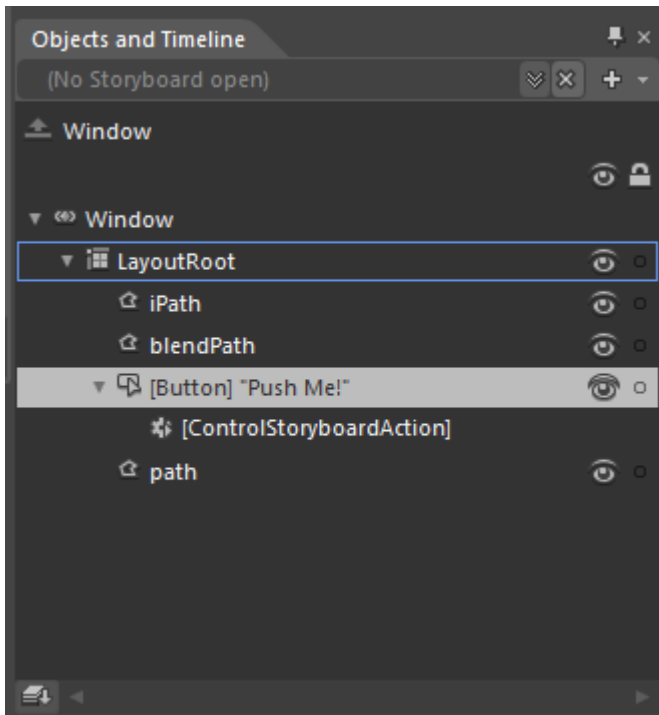


Figure 1.4 The Objects and Timeline panel helps visualize and navigate XAML hierarchy.

1.4.4 The Artboard

The large section in the center of Blend is the Artboard and is the primary focal point of Blend. The Artboard serves many crucial functions. In addition to showing the design in progress, it is instrumental in laying out application containers and controls and it also provides a way to graphically select and edit elements. Like Visual Studio, the design surface can be viewed in three ways: Design View, XAML View, and Split View. Figure 1.5 shows the Artboard configured in Split View mode.

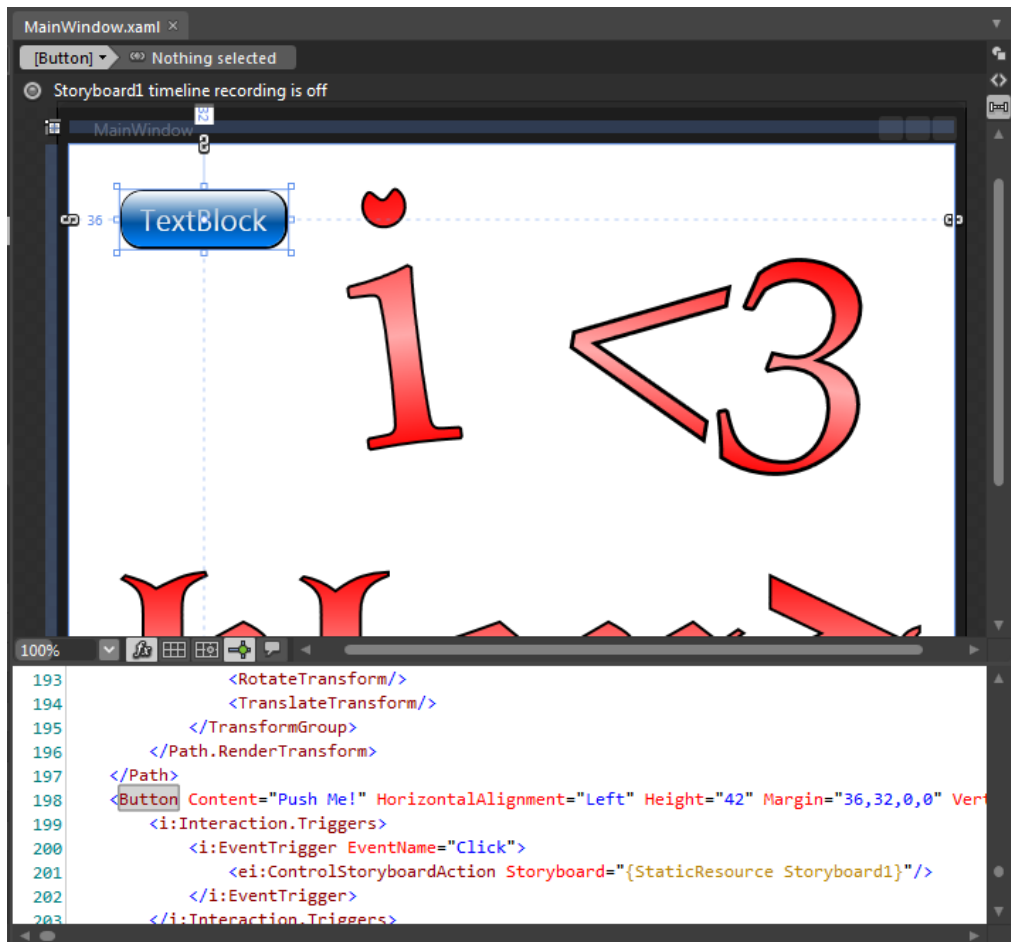


Figure 1.5 The Artboard, shown here in split view mode, is the central focus of design work and document editing.

1.4.5 The Properties panel

On the right side of Blender is another tab control. The first of these is the Properties panel. As with the Projects panel, if one of the other tabs, either *Resources* or *Data*, is selected, just select Properties to see its contents. It may appear empty depending on the state of Blender but we'll cover that in more depth in the next chapter. If it is, click on any element on the Artboard and it should show some content. For now the important thing is to recognize that the function, if not the form, should be immediately familiar to Visual Studio users. As you can see in figure 1.6, the panel is full of editable properties for the selected element.

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=730>

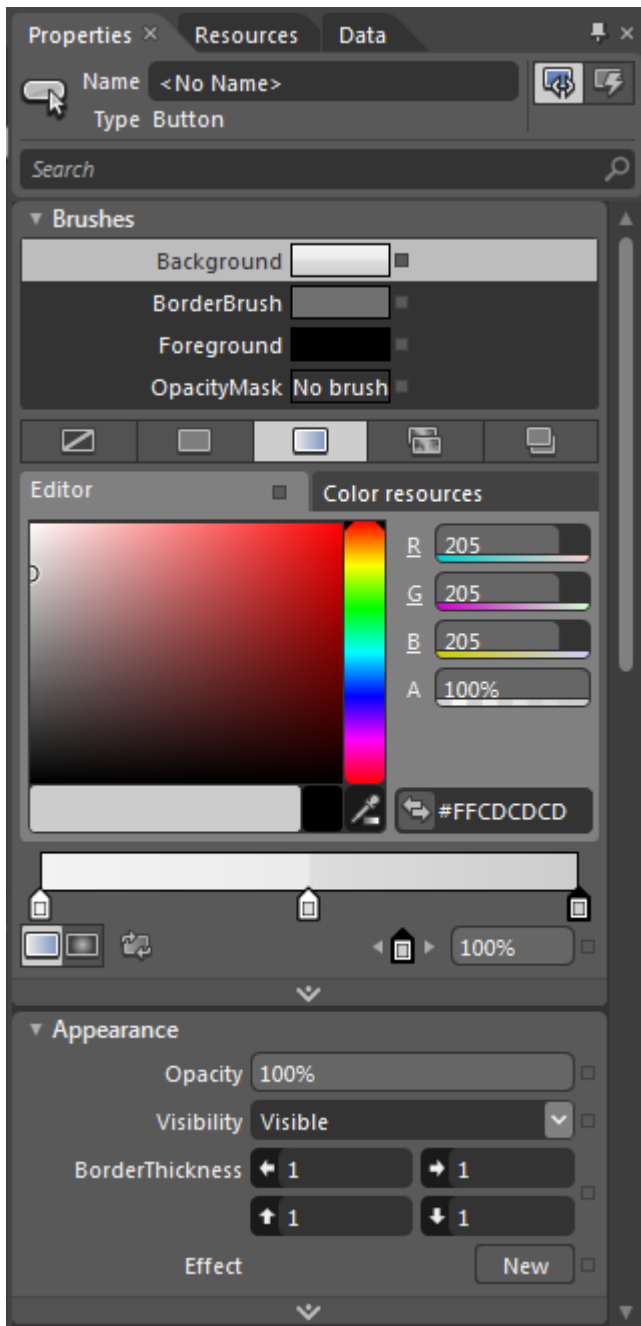


Figure 1.6 The Property panel in Blend serves the same function as the Properties panel in Visual Studio.
 ©Manning Publications Co. Please post comments or corrections to the Author Online forum:
<http://www.manning-sandbox.com/forum.jspa?forumID=730>

Now that you've had the 10,000 foot tour it's time to see Blend in action. Let's dig in and create a project in Blend.

1.5 Hello Blend: your first project

No introductory text would be complete without a "Hello X" project to kick things off! Along the way we'll add content to the Artboard and see firsthand how to use Blend and Visual Studio in tandem. Since there's no time like the present, let's get to work and create your first project in Blend.

1.5.1 Creating a new project

When you first open Blend you should see a Welcome Screen with a "New Project" button (see figure 1.7). If you already have Blend open you can go the traditional menu route File > New Project (Ctrl-Shift-N). Either way you will arrive at the New Project Dialog shown in figure 1.8. As we'll see you can create either WPF or Silverlight applications in the same fashion.

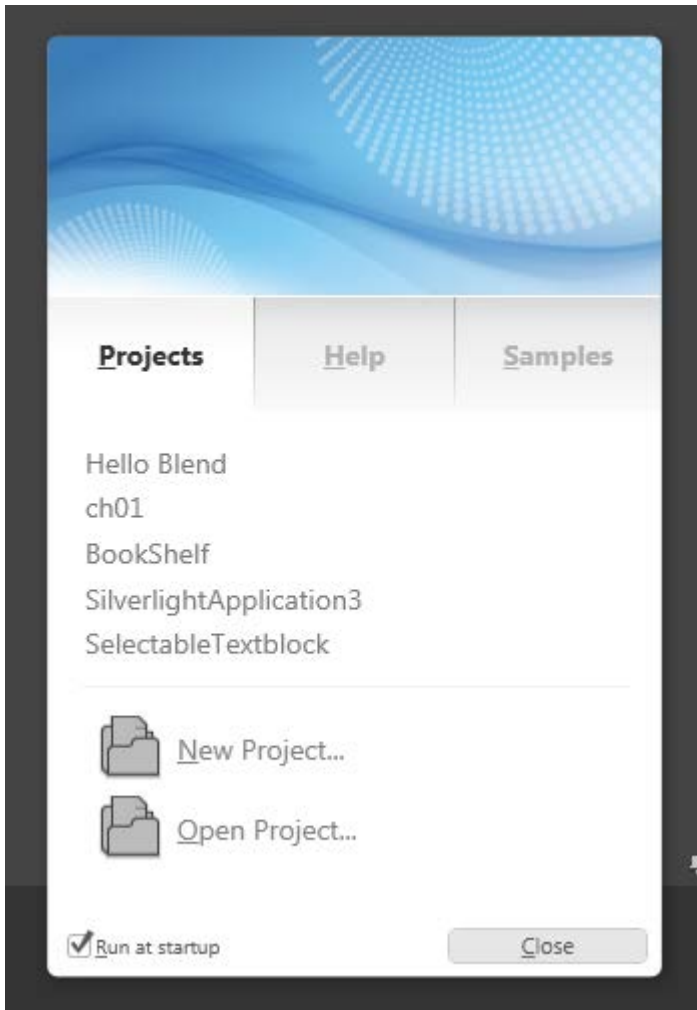


Figure 1.7 Blend's Welcome Screen

CREATING A SILVERLIGHT PROJECT

In figure 1.8 you can see I have several Project Types on the left. When you select one its associated templates are shown on the right. If you've been using Visual Studio for a while this concept should be very familiar to you. For this exercise I am selecting "Silverlight Application + Website". This will generate two projects in my Solution, one for developing Silverlight and an ASP.NET web project for testing, displaying, or deploying the Silverlight application.

Your options may vary depending on the tools and templates you may have installed, so if they don't look exactly like mine don't worry. I've left them visible to illustrate that Blend is extensible, just like Visual Studio. Name the project "Hello Blend" and select the Location, Language, and Version like in figure 1.8, and press OK.

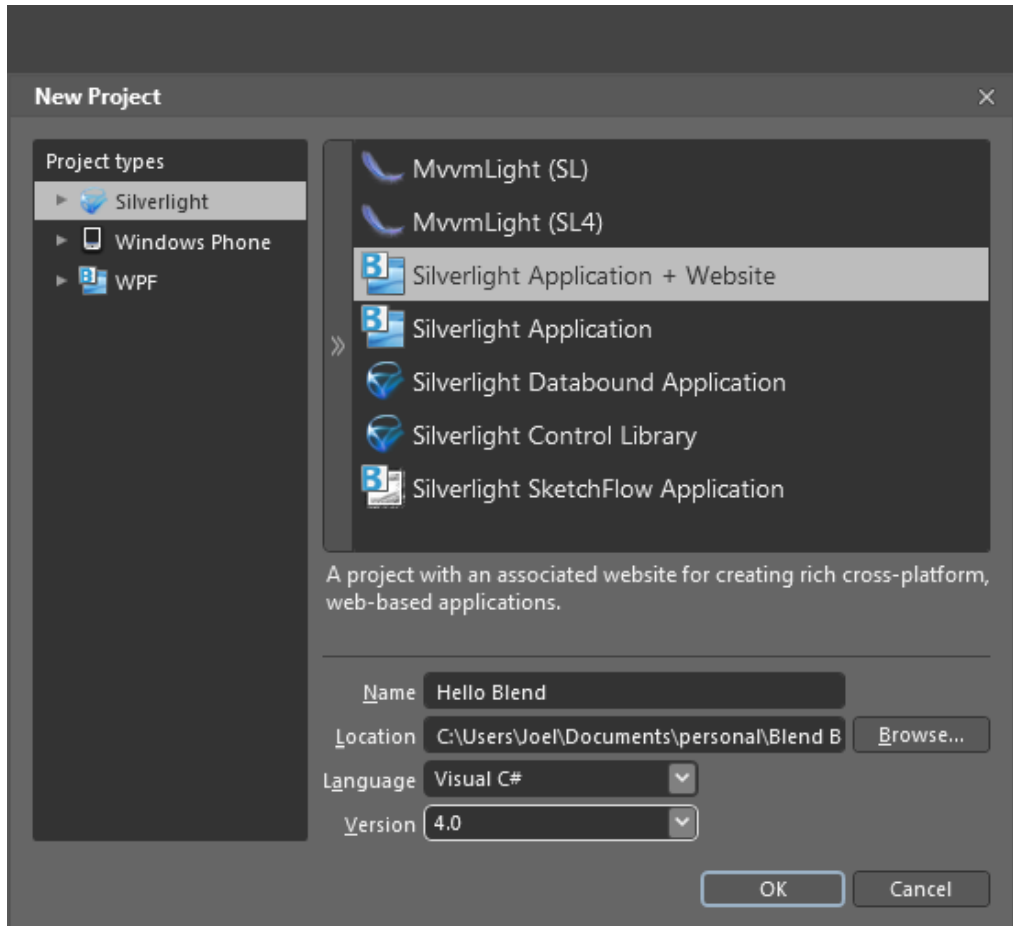


Figure 1.8 New Project Dialog Box, shown here with Silverlight selected.

Congratulations, you've just created your first Silverlight project in Blend!

CREATING A WPF PROJECT

If you are more interested in WPF than Silverlight, feel free to create a WPF application instead. Once we get past the project creation stage the rest of the demo will function the same way regardless of which one you select. In fact, the vast majority of features and tools

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=730>

we'll be using Blend work equally well in both WPF and Silverlight. When something is specific to one or the other I'll be sure to point it out. Figure 1.9 shows the New Project Dialog for WPF.

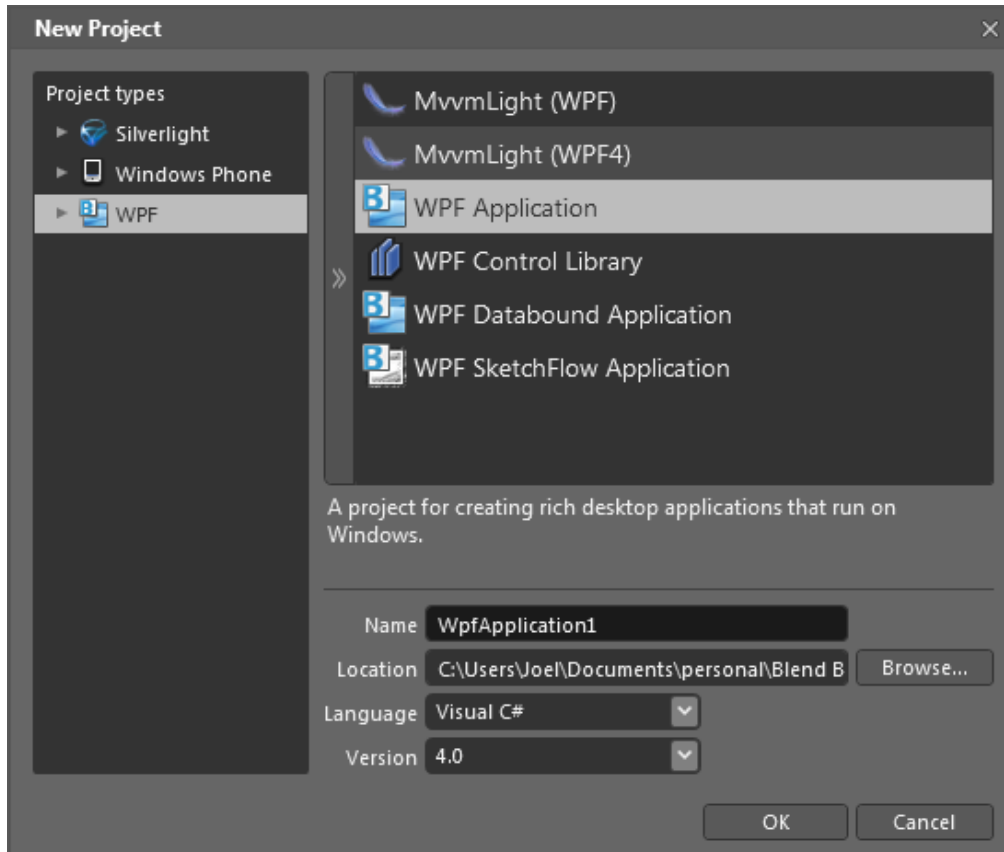


Figure 1.9 The New Project dialog for creating a WPF project.

Remember that WPF and Silverlight are very similar, so whether you created a WPF or Silverlight application, the rest of the demo will be the same. Now that you've created your first project let's add some content.

1.5.2 Adding content to the Artboard

Blend should now look something like Figure 1.10, what I consider the *Tabula Rosa* of Blend.

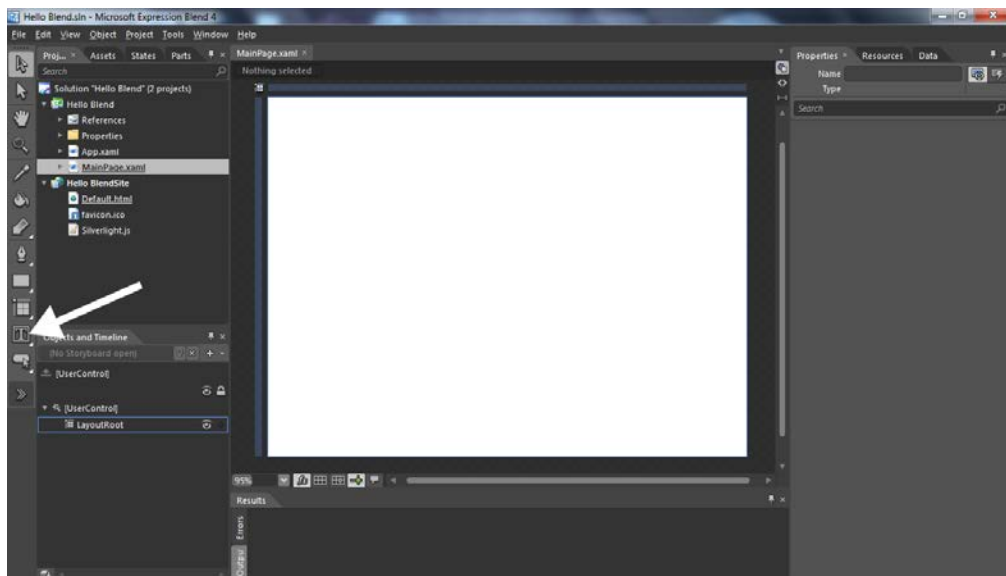


Figure 1.10 A new project in Blend

On the left hand side you'll see I've marked the TextBlock control. Double click that and a TextBlock control will appear in the upper left hand corner of the Artboard. Press Escape to get out of text editing mode and then the letter "V" on the keyboard to return to the Select tool. Now you can grab the TextBlock and drag it wherever you like on the Artboard. Don't worry if this doesn't make sense or seems strange to you, we'll be covering these details later in the book. For now I just want you to get some text on the screen so we can demonstrate using the two tools together. Blend should now look something like figure 1.11.

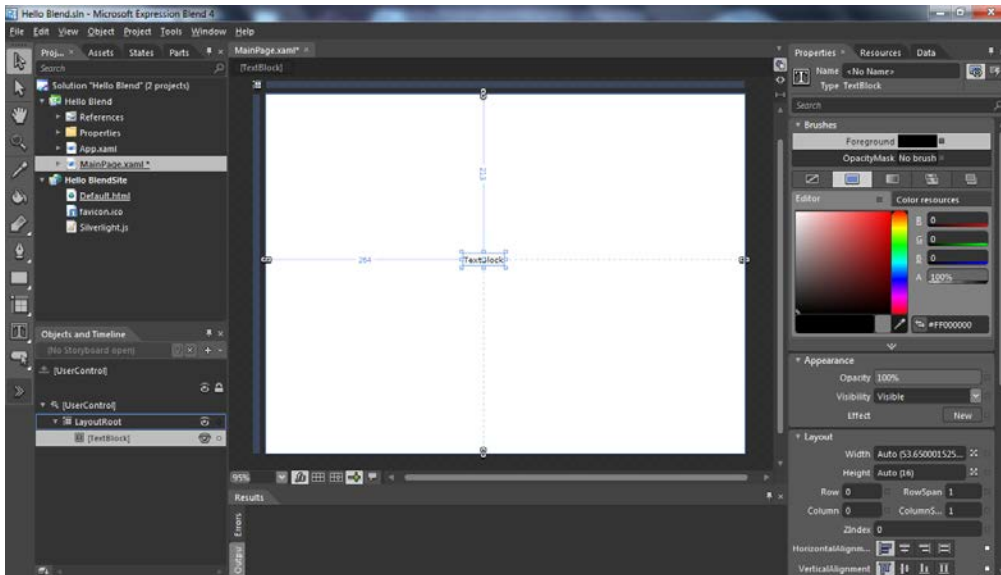


Figure 1.11 The design surface with a TextBlock added.

1.5.3 Moving between Blend and Visual Studio

In this project we're going to use both Visual Studio and Blend. It's definitely overkill for this project, but you need to become comfortable with the crucial task of keeping the two synchronized. To see this in action we'll switch from Blend to Visual Studio, edit some XAML in Visual Studio, switch back to Blend, make some more changes, and finally switch back to Visual Studio.

SWITCHING TO VISUAL STUDIO

Now that we have our project in Blend we need to open it in Visual Studio. The easiest way to do that is to right-click the Solution in the Projects tab (see figure 1.12) and select "Edit in Visual Studio".

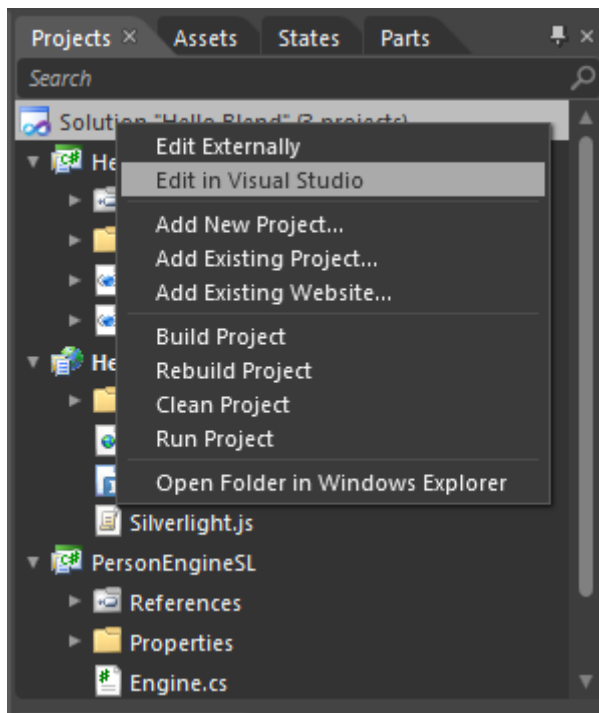


Figure 1.12 Right-click the Solution to open it in Visual Studio

As expected, doing so will open the solution in Visual Studio. You'll see a few messages and screen flashes while Visual Studio is loading. The larger the project the longer it takes to open, but you only experience this delay on the initial startup. I typically go ahead and open the Solution in both tools when I first start working on a project, so I am not held up moving from one to the other while I'm working.

EDITING XAML IN VISUAL STUDIO

Now that Visual Studio is open, find the "MainPage.xaml" file in the "Hello Blend" project and open it so we can edit the XAML in Visual Studio. Please note that this is solely for the purpose of demonstration: I rarely edit XAML by hand.

Find the <TextBlock ...> tag and change its Text property to "Hello Blend". Your XAML should now look something like listing 1.2

Listing 1.2 The XAML with the TextBlock's updated Text property

```
<UserControl
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  x:Class="Hello_Blend.MainPage"
  Width="640"
```

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=730>

```

        Height="480">
<Grid x:Name="LayoutRoot"
    Background="White">
    <TextBlock HorizontalAlignment="Left"
        TextWrapping="Wrap"
        Text="Hello Blend"
        VerticalAlignment="Top"
        Margin="264,213,0,0"
        FontSize="29.333" />
</Grid>
</UserControl>

```

Any time we make changes in one tool and switch to another we need to save our changes. While this process works with either saving or building, my preference is to build the Solution anytime I am switching from one tool to the other. Blend uses the same build shortcut as Visual Studio, Ctrl+Shift+B, so reflexively it an easy thing to do. I adopted this because I would rather deal with build errors in the here and now in the tool that most likely produced them. It limits confusion and unnecessary toggling later.

XAML settings in Visual Studio

In the case that I do need to edit XAML in Visual Studio, there are a couple of settings I use to make it a more palatable experience. First, go to Tools > Options > Text Editor > XAML > Miscellaneous and check the box for "Always open documents in full XAML view." When you open a XAML file with this setting it should go directly to the XAML view and automatically collapse the Design split view.

Next, in order to more easily see the properties of XAML tags, go to Tools > Options > Text Editor > XAML > Formatting > Spacing and under Attribute Spacing select "Position each attribute on a separate line". As a personal preference I also check "Position first attribute on same line as start tag". Now, to reformat the code you simply use the "Ctrl+K+D" keyboard shortcut in Visual Studio (this trick does not work in Blend).

Unfortunately, Blend will not respect these settings. Any time Blend edits the XAML it will lose this formatting, so when you open a XAML file in Visual Studio you will need to reformat it again to see the effects. You can do this very quickly by using the Ctrl+K+D shortcut. You will find the XAML much easier to read and edit this way.

SWITCHING BACK TO BLEND

Now either Save or Build your solution and return to Blend. When you do, you will receive the prompt listed in figure 1.13. I can honestly say I have never had a case where "No" was the correct answer, so press "Yes".

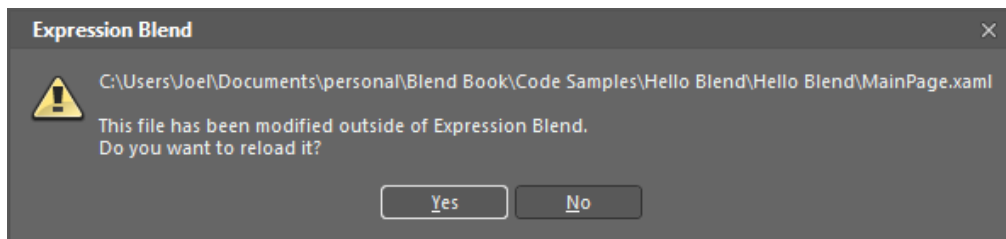


Figure 1.13 Blend prompts you to keep solution files in sync.

You should now see the new text reflected in Blend.

AND BACK TO VISUAL STUDIO AGAIN...

While we still have Visual Studio open, let's see it in action going the other way. Go ahead and make any change to the `TextBlock` you want, any change at all. Don't worry, we'll be covering how to do this in detail later in the book, but for now feel free to experiment a little. Once you've made your change in Blend, go ahead and Save or Build, and then switch back to Visual Studio. When you do Visual Studio will prompt you as seen in figure 1.14. Again, I've never needed to pick an option other than "Yes", and just to speed things up I always select "Yes to All".

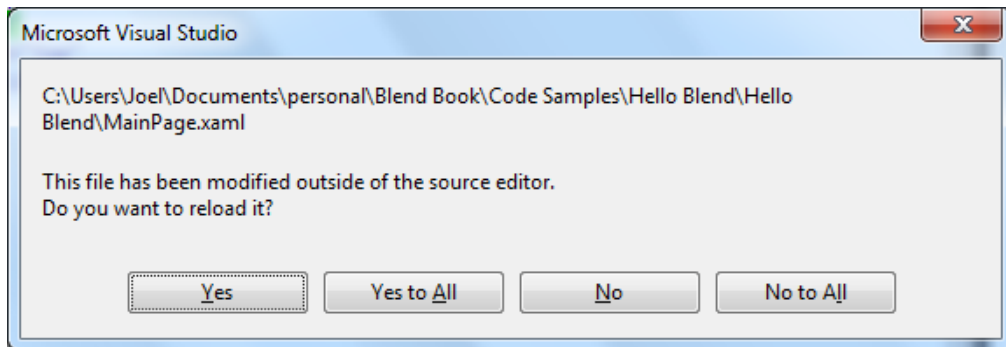


Figure 1.14 Visual Studio's file synchronization prompt

Now that we've seen it in action a few times you should feel comfortable switching from tool to tool. Feel free to experiment and play around some. When you are finished, go ahead and close both Blend and Visual Studio: we'll be starting a fresh project in the next chapter.

1.6 Summary

Users demand experiences that are both functional and visually pleasing and creating those interfaces is a challenging and time consuming task. Blend saves you time by letting you

©Manning Publications Co. Please post comments or corrections to the Author Online forum:

<http://www.manning-sandbox.com/forum.jspa?forumID=730>

craft those user interfaces quickly. Blend makes you more productive and your applications more appealing, a true win-win!

While Blend may appear dark and mysterious it has many similarities to Visual Studio. Learning the details of the related features such as project creation, project management, and properties will help you quickly get comfortable with Blend's interface.

It is critical that you understand that Blend is a companion to Visual Studio, not a replacement. The two are used in a complementary manner and you should select the right one for the task at hand. Blend is often touted as a designer's tool, but I contend that it is a tool for design that developers can and should use to improve their applications.

In the next chapter we'll take a deeper dive into Blend. We'll configure Blend to best suit developers and we'll learn how to navigate applications and the visual tree. You'll also find some tips and tricks for using Blend that will make you lightening fast.