Brennon Williams

In Full Color

Microsoft® Expression Blend® 4

UNLEASHED

SAMS

Brennon Williams

Microsoft® Expression Blend® 4

UNLEASHED

SAMS | 800 East 96th Street, Indianapolis, Indiana 46240 USA

Microsoft® Expression Blend® 4 Unleashed
Copyright © 2011 by Pearson Education

ISBN-13: 978-0-672-33107-7
ISBN-10: 0-672-33107-1

Library of Congress Cataloging-in-Publication Data is on file

Printed in the United States of America

First Printing, April 2011

Editor-in-Chief
Greg Wiegand

Executive Editor
Neil Rowe

Development Editor
Mark Renfrow

Managing Editor
Kristy Hart

Project Editor
Andy Beaster

Copy Editor
Water Crest Publishing

Indexer
Lisa Stumpf

Proofreader
Jennifer Gallant

Technical Editor
Bruce Johnson

Publishing Coordinator
Cindy Teeters

Book Designer
Gary Adair

Composition
Nonie Ratcliff

Contents at a Glance

Table of Contents

About the Author

Brennon Williams is the Chief Technology Officer for the award-winning UK design and development agency, Splendid. Brennon is also a technical advisor to several companies around the world, specializing in the implementation of designer/developer collaboration and workflows.

For almost two decades, Brennon has worked in several countries as a consultant software developer and technical advisor.

Owner of the ExpressionBlend.com website which is due to be launched in 2011, Brennon's technology insights can also be followed at his personal blog, brennonwilliams.com

Brennon was awarded a BS in Computer Science from NYU and has received the Microsoft MVP award for Expression Blend 2008, 2009, 2010, 2011.

Dedications

To my family, friends, and loved ones.

You probably still don't know what it is that I do.

That's OK.

Just know that I love you all and feel truly blessed to have you in my life.

Note to self…

Get some sleep, idiot.

4:18 AM. February 14, 2011.

Acknowledgments

I feel really happy about this book.

That's pretty important, and I hope that comes through as you read through the book.

Most of the time, authors will say writing a book is a killer, and although it's been a tough endeavor, I have to say that my editor at Pearson (Neil Rowe) is one of the reasons why I have managed to do it again.

Neil understands me; he understands that I am very time poor and that I wanted to get this book done as soon as possible, even though it sometimes isn't a schedule that I can stick to, because of the unknowns.

Neil and I agreed to write this book on nothing more than a handshake after a dinner in Las Vegas in 2010 (that reminds me… I need to sign the contract). He understood that I needed to rewrite this topic from scratch because it means more to me than just punching out some text. For me, this book needs to be "the book" on Expression Blend.

Thank you, Neil, for giving me this opportunity. Blackjack and steak in Vegas again this year?

To the Expression Blend team and Expression Management at Microsoft: Thanks for supporting me and listening to me—the good and the bad.

The speed at which you all reply to my questions is very much appreciated and certainly makes me feel like a valued part of your extended team. I hope this book honors the skill and creativity that you have all injected into Blend.

To Bruce Johnson: Thank you for your patience, waiting for chapters, and navigating the instructions that I have stitched together. Your exacting skill is what keeps me honest and what in the end gives the readers an accurate path to follow.

To all the Pearson staff who came together to make this book possible—the reviewers who make sense of my direction and the editors who guide me to creating a better product. Thank you for your endless help.

To my colleagues and associates, who I am privileged to work with at Splendid and around the world. Thank you for the encouragement, support, and belief, which gives me the confidence to write a book. Cheers to you all!

We Want to Hear from You!

E-mail:	feedback@samspublishing.com
Mail:	Neil Rowe Executive Editor Sams Publishing 800 East 96th Street Indianapolis, IN 46240 USA

Reader Services

Introduction

There are many ways that Expression Blend can be described—who it is aimed at and how it should be used.

My view will be different from the next guy or gal, and the one after that. All I know is that I have used this tool almost on a daily basis for several years now on every platform that it supports, and it just keeps getting better and better.

User experience and interactive design is an ever more important part of the solution creation life cycle and the richer the platforms become that Expression Blend supports, the greater the need for this tool, which can assist that process to be imparted into the production solution, working from initial sketches, wireframes, and prototypes, and then through to production implementation.

Expression Blend 4 represents a real step forward in providing designers and developers with the ability to collaborate, as well as to help developers who need a more robust design tool (other than Visual Studio) when working with user interfaces. Blend has matured to the point that very little, if any, code or XAML scripting knowledge is required for the majority of tasks that it allows you to perform, and this book aims to show you how to work in that manner in clear and as much as possible non-technical language.

What Will You Learn in Expression Blend 4 Unleashed?

As you would expect, you will learn how to work with the core features of Expression Blend 4, the creation of compositions, and the structure of solutions in a generic setting.

By far, the most important features of Expression Blend to learn are the combination of several intrinsically connected concepts, as follows:

- Styles and templates
- Parts
- States
- Behaviors
- Animations
- Resources
- Data

Understanding these features collectively will open up the world of user experience design and interactive development on the .Net platform to you. There are many other features to embrace with .Net and indeed with Expression Blend 4, but these concepts specifically are the key areas in terms of UI implementation.

If you are new to Expression Blend, you will find that you can read this book from start to finish and build your knowledge step by step, from beginner concepts through to complex interaction.

The Topics Covered in Expression Blend 4 Unleashed

Expression Blend 4 Unleashed contains detailed instruction and discussion around the core functionality of the tool.

The topics covered are:

- Exploring the Expression Blend interface
- How to work with common properties of user interface elements and controls
- How to work with dynamic layout support
- How to create, edit, and manage Styles, Templates, Parts, States, and Behaviors
- How to work with data in your user interface
- How to apply animations and visual effects
- How to work with advanced controls
- How to work with resources and assets
- How to use Blend to build solutions for Windows Phone 7

How This Book Is Structured

The book is written in a very explicit order, attempting to cater for as many readers as possible.

If you need a quick reference to refresh your understanding of a given topic, you should be able to find the chapter(s) that contains the correct content by logically reviewing the Chapter titles.

Even if you are a veteran WPF or Silverlight developer, if you jump straight into trying to work with Data in Blend, you will become frustrated at both the tool and this book, guaranteed.

Designers should not assume that just because Blend looks similar to other tools one may have previously used in the past, that Blend will work the same. Often, it is simply not the case.

Developers should be aware that Blend is not about XAML. Yes you can edit XAML script in Blend (and I concede that in some advanced cases you will need to do that), but in general, working with Blend to drag and drop, draw and define elements, is lightning fast by comparison of hand cranking the XAML.

There are few shortcuts to working with Expression Blend efficiently, so you should work through this book sequentially. After a few of the early chapters, you will start to gain familiarity with the discussion points and topics of the chapters and once you have traversed the initial learning curve of Expression Blend, you will find that it is quite an intuitive tool to use.

The best part is that you will be able to transfer your skills across platforms from WPF to Silverlight and beyond.

Sample Applications Covered in This Book

Most of the samples in this book are authored for the Silverlight web platform that, combined with the introduction of the Windows Phone 7 Silverlight platform, gives the greatest reach of your potential skill. You should find it easy to work from one to the other at any stage, as well as with WPF and Surface, should they be your platform of choice.

I have written the book by taking the view that you know your way around a Windows PC—you understand where File Explorer is located, and you are comfortable finding images and media files.

What I don't assume is that you have any prior knowledge of .Net, C#, Visual Studio, or any other specific platform package, such as Silverlight. Even if you have done little more then start Windows on a PC, you should be able to work through this book comfortably and with minimal stress.

Before You Begin with Expression Blend 4 Unleashed

It is essential that you have downloaded Expression Blend 4 or Expression Studio. It will also help if you have a version of Visual Studio installed. All those packages can be found by going to Microsoft's website and searching for the relevant download.

CHAPTER 2

Discovering the Expression Blend Interface

IN THIS CHAPTER

- Customizing the interface
- Working with the artboard
- Working with the code and XAML Editors
- Working with Objects and Timelines
- Working with the Tools
- Working with Properties and Events
- Working with Resources
- Working with the Data panel

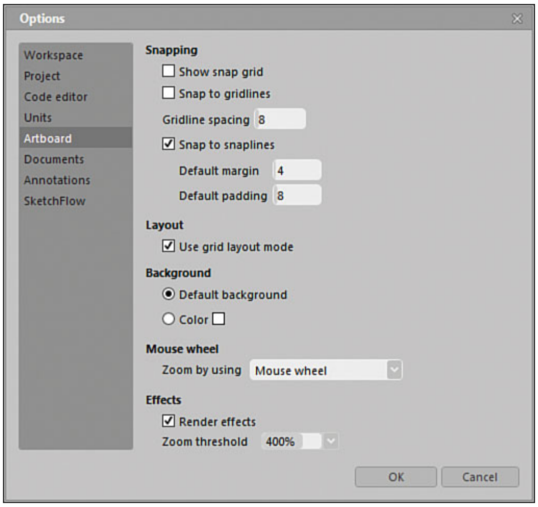
The Blend UI has changed dramatically from the first version, but there have been a lot of incremental changes that have evolved each step of the way. Expression Blend 4 has the fewest new editions to the UI directly of any release to date, but it still has some secrets that are waiting to be let out of the box.

In this chapter, you learn about the core parts of the Expression Blend UI, understanding the key changes to the user interface (which is dependent on project type) and focusing on those panels that provide the functionality that you will use most moving forward.

The Expression Blend Interface Theme

Amazingly, one of the first questions that many users new to Expression Blend ask is: “Can I change the UI theme?”

The short answer is yes—you have a choice of two themes, “Expression Dark,” shown in Figure 2.1, and “Expression Light,” shown in Figure 2.2.



Options

Workspace
Project
Code editor
Units
Artboard
Documents
Annotations
SketchFlow

Snapping

☐ Show snap grid
☐ Snap to gridlines
Gridline spacing: 8
☑ Snap to snaplines
Default margin: 4
Default padding: 8

Layout

☑ Use grid layout mode

Background

● Default background
○ Color ☐

Mouse wheel

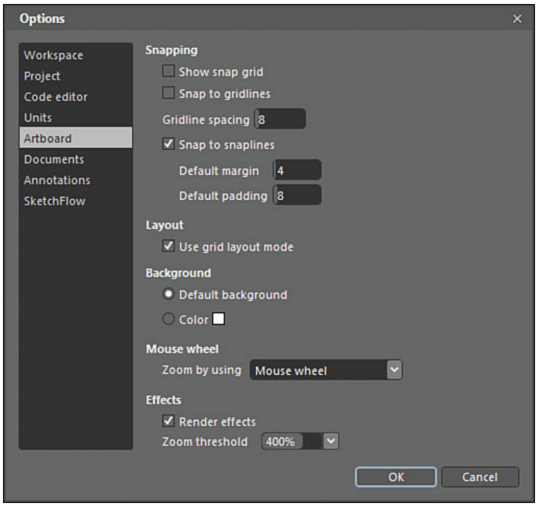
Zoom by using: Mouse wheel

Effects

☑ Render effects
Zoom threshold: 400%

OK Cancel

FIGURE 2.1 An example of the Expression Blend Light theme.



Options

Workspace
Project
Code editor
Units
Artboard
Documents
Annotations
SketchFlow

Snapping

☐ Show snap grid
☐ Snap to gridlines
Gridline spacing: 8
☑ Snap to snaplines
Default margin: 4
Default padding: 8

Layout

☑ Use grid layout mode

Background

● Default background
○ Color ☐

Mouse wheel

Zoom by using: Mouse wheel

Effects

☑ Render effects
Zoom threshold: 400%

OK Cancel

FIGURE 2.2 An example of the Expression Blend Dark theme.

There has been several requests over the years to enable the user to customize the theming of the Blend interface, but as of yet, it hasn't been a high-enough priority of the production teams to include this feature.

During this chapter, you see the Options panel, which enables you to modify a lot of settings to suit your needs and, of course, switch between the two themes and change the background color and fonts used.

As you can clearly see from Figure 2.3, the interface in its default state can look a little overwhelming and uninviting at first. Most designers probably would feel more at home compared to developers, but after a short time using the application, you will no doubt change it to provide a more natural flow for your eyes.

You Have Two Choices

I must admit that I am yet to see anyone (other than myself) use Expression Light in normal day-to-day usage, but it's a personal choice. I change it mainly to fool myself into thinking that I am doing something different every once and a while.

NOTE

Why Only Two Themes?

In earlier versions of Expression that were being trialed by insiders, there were several themes available, but testing of the users revealed that the dark theme was the most heavily used—thus the default setting of Expression Dark.

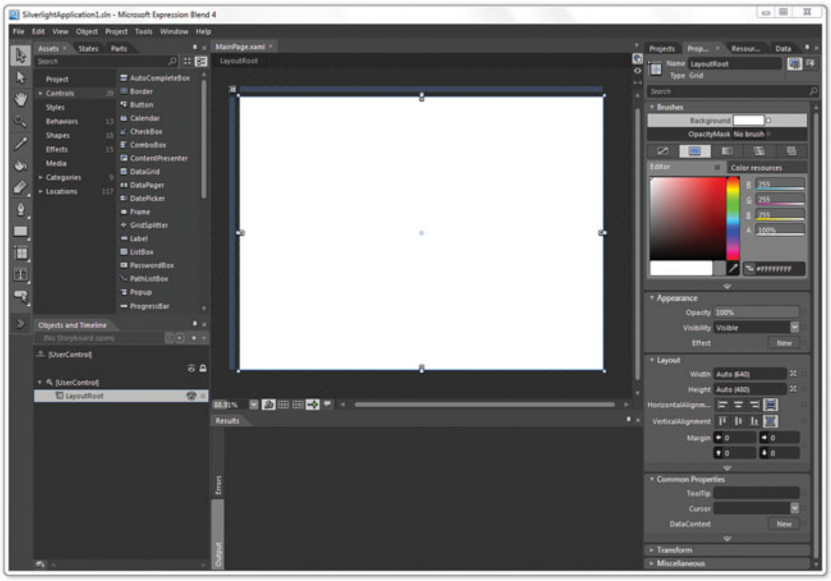


FIGURE 2.3 The general default Expression Blend 4 user interface.

Focus on the Artboard

The decision to create this dark theme and panel layout was born from the idea around directing the users' focus to the design surface (a.k.a. the artboard) and allowing it to stand out. This theory is not uncommon in a lot of high-end design applications, and after using the application for around 50 odd hours straight at times (CES 2010—Venetian Hotel, room 1210, Las Vegas), I find the usability still very comfortable, although by that stage I am usually hallucinating anyway.

How the Experience Changes

The user interface and indeed the user experience changes in the product depending on the project type that you select. There are several panels that become exposed or hidden depending on their relevance to the task at hand.

In the following sections, you are going to have a quick look at those changes so you are not surprised when you change project type later in the book and wonder where a certain panel has gone. First, however, it will be best for you to familiarize yourself with the most common panels that are almost always present. At the end of this chapter, you take a quick look at the Options panel and customize your experience even further.

Common Panel Framework

The Expression Blend common panel framework is very flexible in terms of how you can position and move panels and tabs to suit your working style.

In the following steps, you will create a new project and learn about the options available when performing this function.

1. Open Expression Blend, and on first use, you will be presented with a dialog similar to that shown in Figure 2.4.
2. Close this dialog. You look at this in detail in Chapter 3, "Using Expression Blend for the First Time." Don't be concerned if this dialog isn't showing at present, though.

   At the top of the Blend UI, you see the main menu system providing many options of which you are interested in the File option in the lefthand corner.
3. Select "File" and then select "New Project..." from the drop-down menu items.

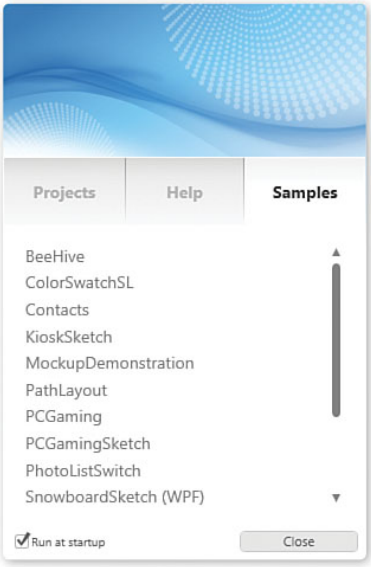


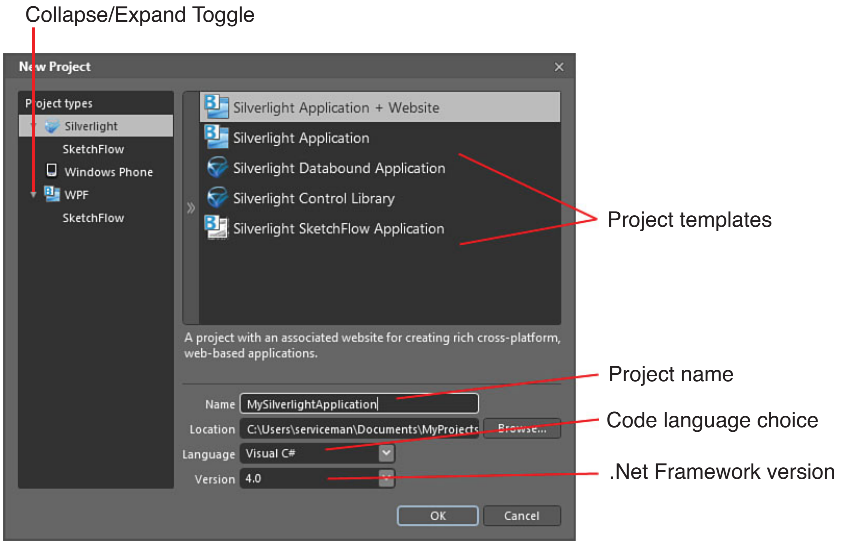
FIGURE 2.4 The default start-up dialog for Expression Blend.

Menu Instructions

From here on out, when being instructed on a specific menu sequence you will be guided by arrow format describing the sequence of menu items to select as the following example shows:

File->New Project…

You will now see the New Project dialog, as shown in Figure 2.5.



A project with an associated website for creating rich cross-platform, web-based applications.

Name MySilverlightApplication
Location C:\Users\serviceman\Documents\MyProjects
Language Visual C#
Version 4.0

FIGURE 2.5 The New Project dialog.

Project Types
With the item highlighted at the top of this list, you see all installed project templates in the right side of the dialog. Subsequent selection of the items “Silverlight”, “Windows Phone” or “WPF” show the specific template(s) included with each project type.

Hide/Close Project Types
Toggle to show or hide the respective template list.

Project Templates
You see varying project templates for you to select, which have specific implications for how your project solution will work and function.

Name
It goes without saying for most people that you should name your solution as descriptively as possible. Try to ensure that you don’t use spaces or strange symbols in your solution name.

Location

You see a Browse button next to the input that opens a dialog enabling you to select the master location of your solution files. Make sure you remember where this is for future use, although there exists some helpers in Blend to return you to this location in Windows Explorer.

You also see in the screenshot of Figure 2.5 that I have created a location specific for the examples I create in this book.

Language Dropdown

You see that you have two language options to choose from in this drop-down; C# and VB.net are both .Net-compliant languages that Expression Blend supports. As mentioned previously, C# is the language that you will be using in this book.

NOTE

Why Use C# Over VB.net?

This is a question that really opens Pandora's Box in many development circles, as people claim one language is better than the other for various reasons. I won't weigh into this argument if I can help it, because fundamentally both languages compile to the same codebase that .Net uses, although there exists several differences, both positive and negative.

The choice is yours, and neither is a wrong or right choice. I began as a VB developer a very long time ago, so I do have a soft spot for it; commercially, however, it is clear that C# is the language of choice for vastly more job opportunities, so why limit yourself?

Version

Expression Blend supports Versions 3.0 and 4.0 of Silverlight and Version 3.5 and 4.0 of the full WPF-based .Net framework; therefore, the choice is simple in those respects. Each version of the .Net framework offers differing functions and features with the later versions offering more choice and hopefully better performance than the predecessor. Expression Blend 1 and 2 do support early versions of the .Net framework before version 4.0, but it must be noted that the project types created by Expression Blend 1 and 2 only associate with Visual Studio 2008.

Expression Blend 4 creates and support Visual Studio 2010 project types.

This book will focus on targeting the .Net framework version 4.0 for both WPF and Silverlight, as well as Silverlight 3.0 + WP7 enhancements for Windows Phone 7.

4. In the project template list, select the Silverlight Application + website option.
5. Leave the default name and, if suitable, location as is provided.
6. Ensure that you have C# selected as your language type.

NOTE

Why Don't You Have the Same Project Types Showing?

Depending on what you have installed, Expression Blend shows different project template options, and this should not concern you at present. In the following sections, you see where to get copies of various Expression Blend add-ins, which give you similar project template options as shown.

7. Ensure that you have version 4.0 selected.
8. Click OK.

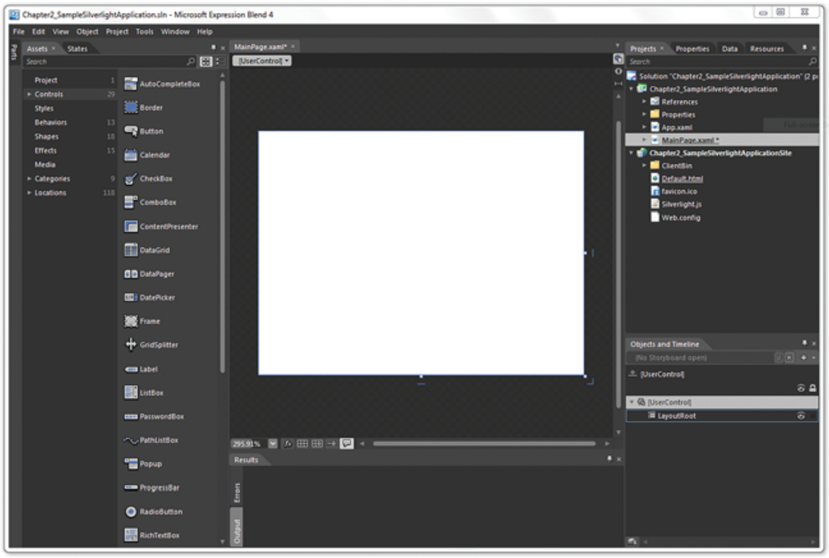

Chapter2_SampleSilverlightApplication.sln - Microsoft Expression Blend 4
File Edit View Object Project Tools Window Help
Assets States
MainPage.xaml*
Projects Properties Data Resources
Objects and Timeline
Results

FIGURE 2.6 The Expression Blend user interface for Silverlight applications.

As shown in Figure 2.6, there is a lot going on in the user interface, as the default Silverlight Application + Website template specifies which panels should be visible.

> **Specifying Which Panels are Shown by Default**
>
> Once you make a change to the panels that are shown, Expression Blend will remember those settings and from there on out, the same panels and layout in the Blend UI should remain in place for all project types.

Moving Panels Around

Figure 2.7 shows an example of the panel and tab collections in Expression Blend, which offer a robust mechanism for customizing your layout and preferred workspace requirements. The movement of the tabs and docking is very similar to that of Visual Studio if you have previously experienced that product.

Tab Collection

Tabs are held in collections in the Blend interface, and you have the options to change the order of the tabs by simply dragging them left and right to re-order them.

Auto Hide

This option enables you to hide/unhide the selected tab collection. You can choose to “float” any tab by click and holding your chosen tab and moving the tab away from the collection. When you move the tab back toward the collection, you will be given a hint of its new placement by a floating blue border, as shown in Figure 2.8.

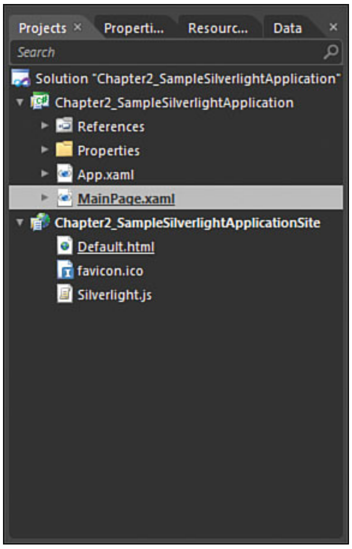

Projects
Properti...
Resourc...
Data
Search
Solution "Chapter2_SampleSilverlightApplication"
Chapter2_SampleSilverlightApplication
References
Properties
App.xaml
MainPage.xaml
Chapter2_SampleSilverlightApplicationSite
Default.html
favicon.ico
Silverlight.js

FIGURE 2.7 The Project/Properties/Resources/Data tab collection.

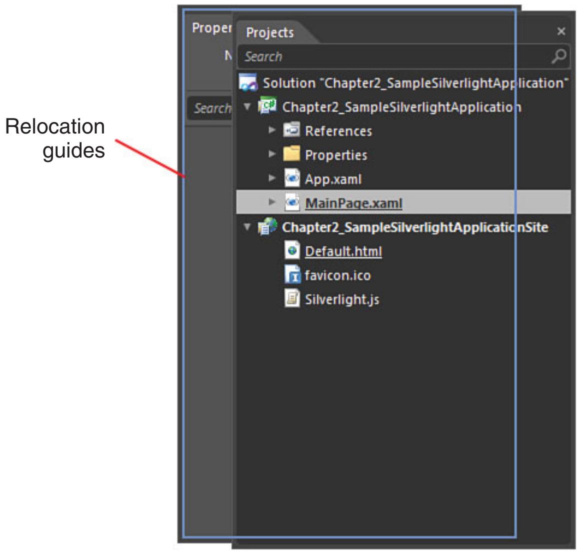

Relocation guides
Proper
Projects
Search
Solution "Chapter2_SampleSilverlightApplication"
Chapter2_SampleSilverlightApplication
References
Properties
App.xaml
MainPage.xaml
Chapter2_SampleSilverlightApplicationSite
Default.html
favicon.ico
Silverlight.js

FIGURE 2.8 How to relocate a Tab panel.

Figure 2.9 shows that the Properties tab has been floated and the rest of the tab collection is now in Auto Hide mode, which you can mouse over any of the hidden tabs to view that specific tab. Click on the pin again on any of the tabs, which will return the entire collection back to its original position.

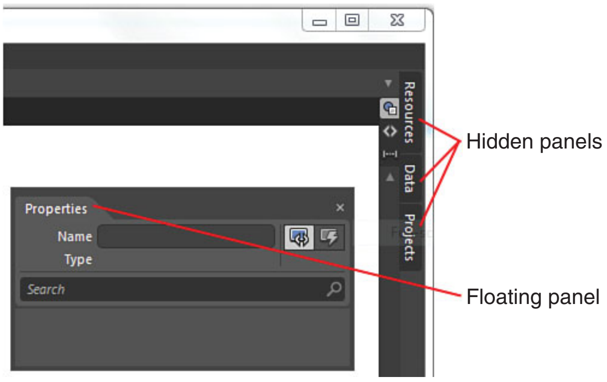


FIGURE 2.9 The collapsed panel collection and the floating Properties tab.

Docking

You can drag a tab item to roughly the location on the screen where you would like it to free float, including an edge of the UI to dock the panel. You should be again presented with the blue border guide that gives an indication of a new docking position, as shown in Figure 2.10.

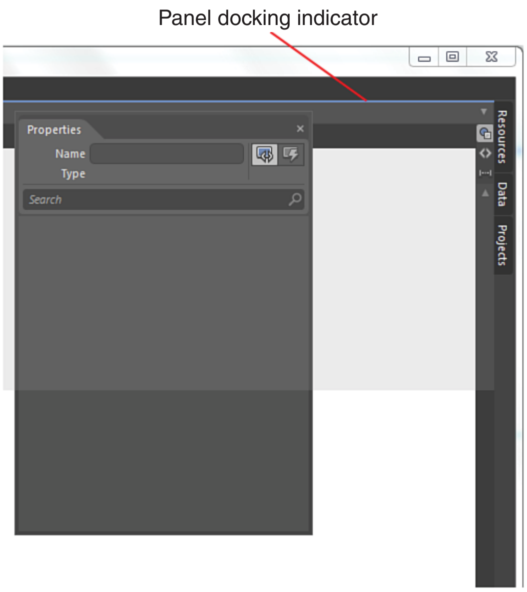


FIGURE 2.10 The indicated position of a docked tab is shown in the interface.

Project Panel

The Project panel represents all the pieces of your solution, including files, resources, settings, and any references to external assemblies and application parts that you may be using. Figure 2.11 shows a breakdown of the Project panel in a typical Silverlight Application + Website project.

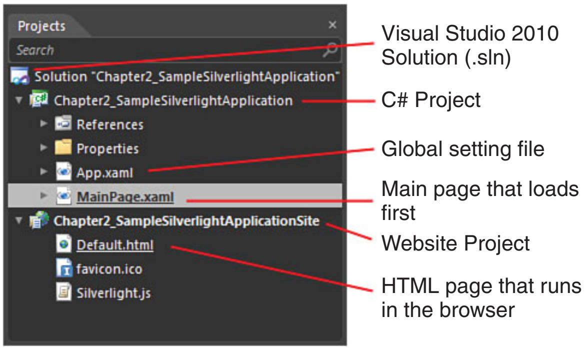

Projects
Search
Solution "Chapter2_SampleSilverlightApplication"
Chapter2_SampleSilverlightApplication
References
Properties
App.xaml
MainPage.xaml
Chapter2_SampleSilverlightApplicationSite
Default.html
favicon.ico
Silverlight.js
Visual Studio 2010 Solution (.sln)
C# Project
Global setting file
Main page that loads first
Website Project
HTML page that runs in the browser

FIGURE 2.11 An example of a Silverlight Application + Website project.

You become more familiar with what layouts of varying projects are best as you gain more experience with Expression Blend, and it is important that if this is your first use of the product, you don't allow yourself to be overwhelmed with any perceived complexity.

Solution development is a complex subject that has many variants to understand; therefore, covering such a topic is beyond the scope of this book. You will, however, gain further understanding of some of the more typical usage and creation scenarios of solutions while reading the rest of this book. Specifically, you learn how to create and reference entirely separate projects when using resources, as detailed in Chapter 13, "Skins, Themes, and Resource Dictionaries."

TIP

Can I Customize the Project Types?

There exists several project types that you get out of the box with Blend, but it is also possible to load additional types and to create your own. The methods of doing that is beyond the scope of this book, but a particularly good article on how you might go about performing such a task is located here: http://goo.gl/7HGzc

Expression Blend Options Dialog

The Options dialog in Expression Blend enables you to specify a host of features, as well as some default settings that might help you work more efficiently. You can access this dialog by selecting the Tools menu and then the Options menu item at the bottom.

NOTE

Navigating Menus

From here on out, you will be directed to access menus with the following format: Tools->Options.

Workspace

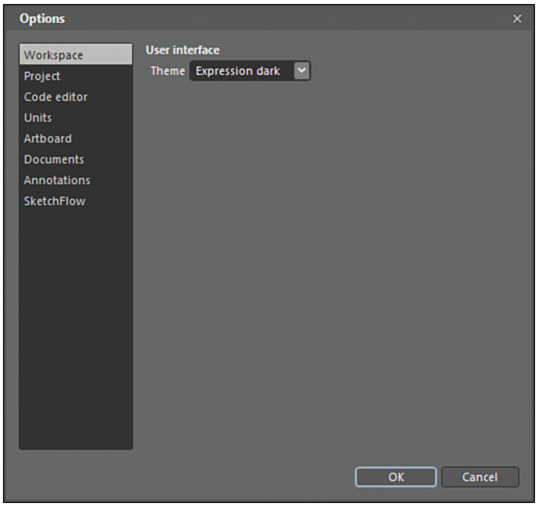
As Figure 2.12 shows, the Workspace options are extremely limited, allowing you simply to select either the Dark or Light Theme.

Project

Consisting of five options shown in Figure 2.13, you have a choice of some self-explanatory settings and a few that are probably a little cryptic to the new user of Expression Blend.

The first option simply enables you to enforce a name on the creation of a new element on your artboard. By default this is off, so when you add a new Button element, for example, you see that it doesn't have a name and appears in the Properties panel as <No Name>. With the option selected, your Button being added would be named "Button," the next will be named "Button1," and so on. It is my advice to leave this option unchecked and to only name elements that you strongly feel need a descriptive title.

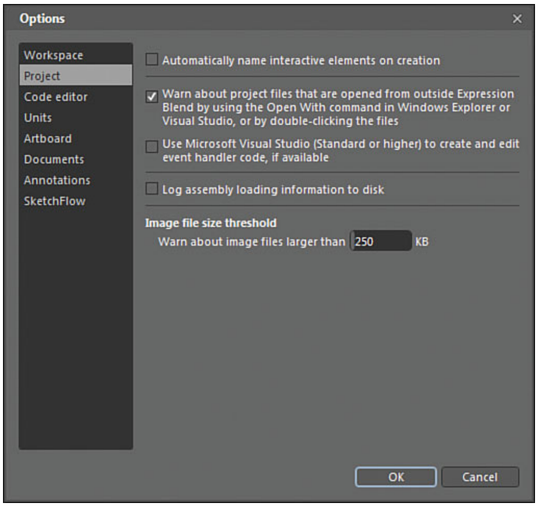
There are many technical reasons as to why you shouldn't name elements but such discussions are outside of the scope of this book.



User interface

Theme: Expression dark

FIGURE 2.12 Expression Blend Workspace options.



☐ Automatically name interactive elements on creation

☑ Warn about project files that are opened from outside Expression Blend by using the Open With command in Windows Explorer or Visual Studio, or by double-clicking the files

☐ Use Microsoft Visual Studio (Standard or higher) to create and edit event handler code, if available

☐ Log assembly loading information to disk

Image file size threshold

Warn about image files larger than 250 KB

FIGURE 2.13 The Project options.

What you should know is that if you intend to do a little bit of coding to interact with elements, you need to have a name for any elements that you want to work with in code. Also, elements that are used in storyboards, for example, are named by Blend if not already done so by you, because the code or markup needs a key reference in order to manipulate the right elements.

The next option is on by default. This is quite handy as a warning system to let you know that a file inside your project has been opened by an external application. This can save versioning issues that can arise from file modification in multiple locations.

The next option is quite explicit about Visual Studio. It states that if you have Visual Studio (Standard or Higher) installed on your machine, you have the option to open and edit code files from within Visual Studio rather than Expression Blend.

As you see shortly, the code editor in Expression Blend is rather limited. It serves a purpose in some respects, but again my advice is that if you have Visual Studio installed, you will get a far better code editing experience in that tool compared to the limited code editor in Blend.

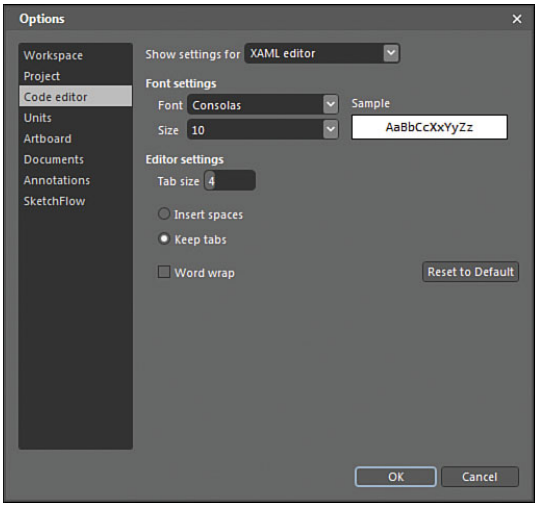
Log assembly loading to disk refers to a Debug assist mechanism that is used when reporting issues to the Expression Blend team. It generates a file that can assist in identifying issues with the Blend application.

Image file size threshold by default is set to 250kb. Allowing Blend to add large files to the output structure of your application can sometimes make it easier to manage your solution package. Typically, you would do this is you know this same file will be used continuously in your solution.

Code Editor

As shown in Figure 2.14, the Code Editor options are quite small. It is interesting to note is that the first option allows you to specify changes for either the code editor or the XAML editor.

These settings are all personal choice, and it's handy to note that if you make a real mess of setting your options such as font and size, you can hit the Reset to Default button to get it all back again.



Show settings for: XAML editor
Font settings
Font: Consolas
Size: 10
Sample: AaBbCcXxYyZz
Editor settings
Tab size: 4
Insert spaces
Keep tabs (selected)
Word wrap (unchecked)

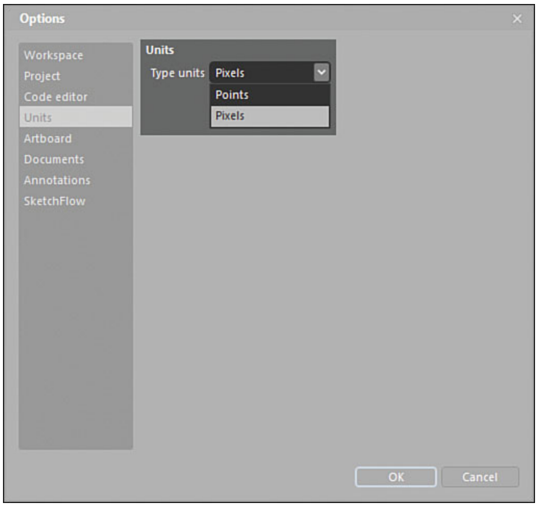
FIGURE 2.14 The Blend Code Editor options.

Units

Specifically referring to Type units of measure, my personal preference is to show pixels here as most of my design software such as Illustrator also uses pixels as a unit of measure for font types. As Figure 2.15 demonstrates, you have the choice of pixels or points.

Artboard

Under the title "Snapping," you see six settings, as shown in Figure 2.16. The first two will be detailed later in this chapter when investigating the Artboard, as these settings can also be turned on and off directly from settings available locally. It is also much easier to explain what the differences are between Snapping to grid and gridlines, as well as the fourth setting of Snap to snaplines.



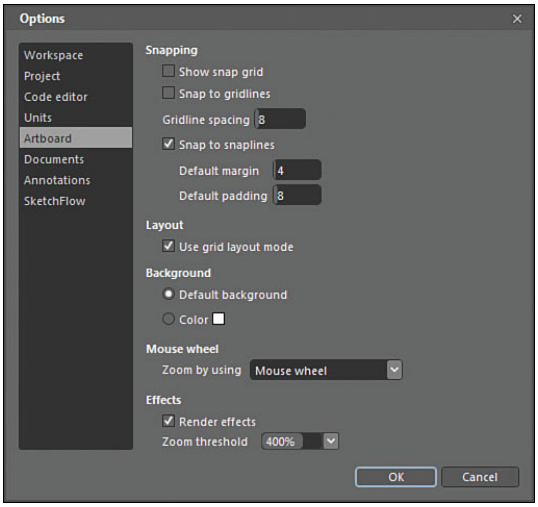
Units

Type units: Pixels

Points

Pixels

FIGURE 2.15 The available unit of measure options.



Snapping

- [ ] Show snap grid
- [ ] Snap to gridlines

Gridline spacing: 8

- [x] Snap to snaplines

Default margin: 4

Default padding: 8

Layout

- [x] Use grid layout mode

Background

● Default background

○ Color ☐

Mouse wheel

Zoom by using: Mouse wheel

Effects

- [x] Render effects

Zoom threshold: 400%

FIGURE 2.16 The Artboard options.

The setting for Gridline spacing should be pretty self-explanatory, but what is important to note here is that the default value shown of 8 is the same value of the Default padding setting and the Default margin value is half of that at 4. You should try to maintain these ratios if you change the settings to see consistency in your snap layout scenarios.

Default margin is shown in Figure 2.17, as the pink area between the two buttons and Default padding is shown as the pink area between the border of the Parent Grid element and the buttons that are sitting inside of it.

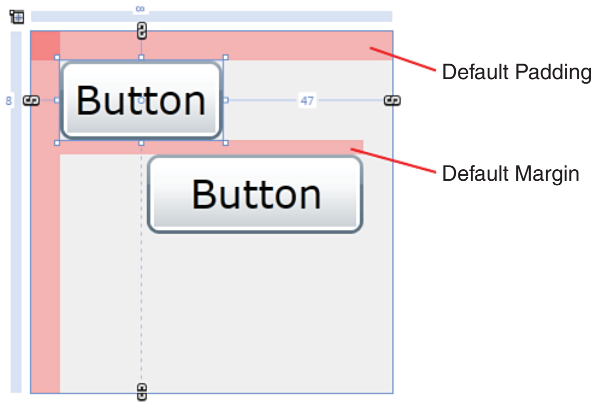


FIGURE 2.17 Default margin and padding settings are applied and shown on the artboard.

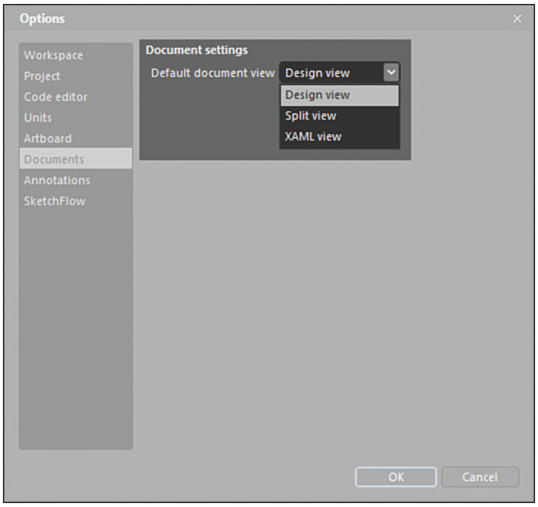
Under the heading of "Layout" resides a single setting, enabling you to set Use grid layout mode. You are advised to ensure this is turned on. More details around the reasons for that will be explained in Chapter 5, "The Art of Layout."

> **NOTE**
>
> **Snap to Snaplines**
>
> For the Default margin and Default padding indicators, you must have Snap to snaplines enabled either in the Artboard options, or toggled on the design surface.

The Background settings enable you to specify a background color for behind the collections of controls you are working on, or you can just stick with the default.

The Mouse wheel settings are again self-explanatory.

The Effects settings might be a little cryptic until you understand what they are referencing and why you can set a threshold value in percentage terms. Effect rendering is discussed in the Artboard part a little later in this chapter. For now, you can leave it at its default of 400%.



Document settings

Default document view: Design view

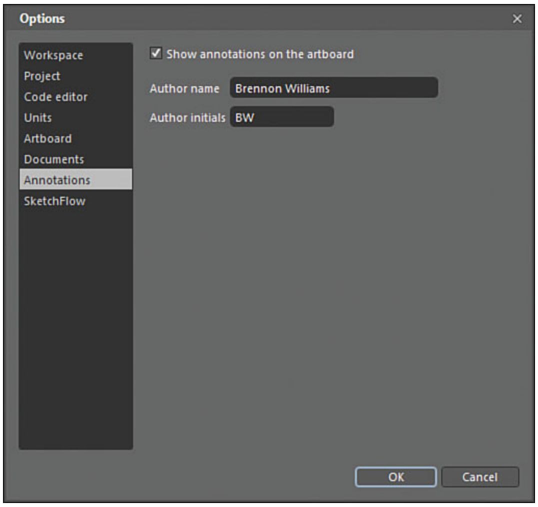
FIGURE 2.18 The settings options available for the Document section.

Documents

Figure 2.18 details the choice you have for setting the default view when opening a document in the artboard. Split view, as you would imagine, splits the view into one-half design surface and one-half XAML view.

Annotations

Annotations (with their options shown in Figure 2.19) are available to post all over your applications and can be extremely helpful in guiding team-mates to areas of concern or specific notes for items as well as indicating to multiple audiences your point of view. More detail will be provided later in this chapter under the design surface, explaining how to create and view annotations.



☑ Show annotations on the artboard

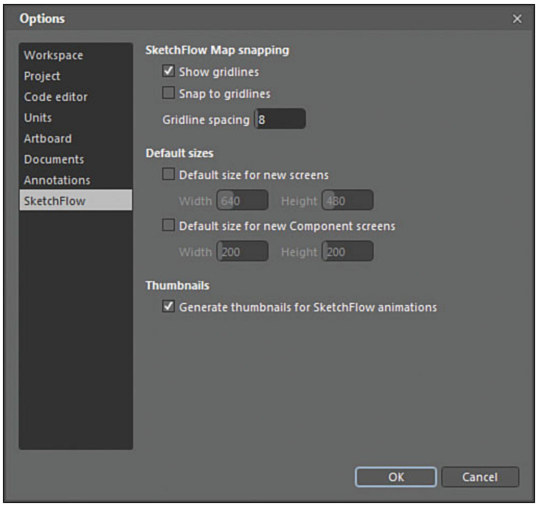
Author name: Brennon Williams

Author initials: BW

FIGURE 2.19 The Annotations options.

SketchFlow

The SketchFlow Options panel shown in Figure 2.20 is really very much out of context in this part of the book but is shown here for completion. For this reason, the specifics of the settings will be detailed in Chapter 9, “Working with SketchFlow.” Most of the settings should be recognizable to you, as they mirror other settings in this section.



SketchFlow Map snapping

☑ Show gridlines

☐ Snap to gridlines

Gridline spacing 8

Default sizes

☐ Default size for new screens

Width 640 Height 480

☐ Default size for new Component screens

Width 200 Height 200

Thumbnails

☑ Generate thumbnails for SketchFlow animations

FIGURE 2.20 The SketchFlow options.

The Artboard

The Artboard is contained within a primary viewing area of the Blend UI that consists of three main display types: the Design Surface, the XAML viewer, and the Code viewer.

The design surface is similar to other WYSYWIG (what you see is what you get) type editors you might have seen before. It is a rich interactive surface enabling you to drag and drop elements onto it from multiple panels, as well as to manipulate the visible elements with ease and simplicity.

Figure 2.21 shows how most elements have what are called “adorners” that allow you to directly modify the elements with your mouse. You can also directly manipulate other properties, such as layout of elements and grouping, and in the case of path elements, you can also use tools to create artwork at will.

The design surface can also be directly interacted with the use of shortcut keys and shortcut keys, plus the mouse as a combination, which Table 2.1 has a breakdown of some of that functionality.

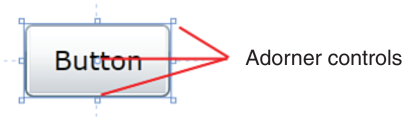


FIGURE 2.21 A sample of control adorner on the artboard.

TABLE 2.1 Handy Keyboard Shortcuts

Function	Keys
Cycle workspaces	F6
Show/Hide all panels	F4
Pan around the workspace	Spacebar and move the mouse
	Hold the Shift key to pan left and right while scrolling
	Select an element in the Objects and Timeline panel and double-click the Pan tool in the toolbar to pan to the selected element
Switch to element selection while keeping the same toolbox element selected	Hold Ctrl key down
Zoom	Mouse wheel
Zoom—fit selection	Ctrl + 0

Figure 2.22 shows some of the functionality attached to the artboard itself. The following gives some detail as to what each function is and how to use them.

Zoom

As you have read in Table 2.1, scrolling your mouse will zoom in and out on the artboard. You can also use some preset values in the Zoom dropdown, shown at the bottom of the artboard space. The last item, "Fit selection," shown in Figure 2.23 can be especially handy and is also documented in Table 2.1. You also note that the toolbar has a Zoom tool that performs similar functions all with just a little more control.

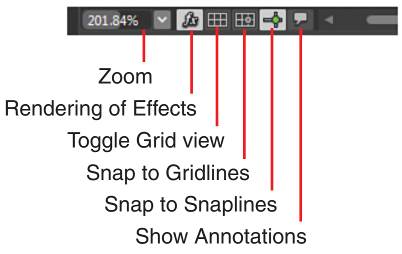


FIGURE 2.22 The attached artboard function controls.

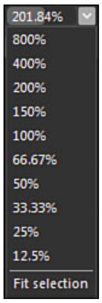


FIGURE 2.23 The Zoom option dropdown.

The View menu item also details several Zoom options, including Actual Size.

Rendering of Effects
As the name suggests, rendering of effects controls if effects (such as Blur and DropShadow) are seen in design time on the artboard. Why would you care about this? The rendering of effects inside the design time preview can be quite expensive in computation cycles, which can reduce performance of Blend considerable, especially when you are zoomed in heavily on items.

For this reason, you also note that previously in the Artboard Options section of this chapter, you could specify a Zoom Threshold. This setting constrains Blend so as it will not render effects regardless of whether you have this option selected or not.

Toggle Grid View
Figure 2.24 shows the Grid effect that is overlaid on your artboard, helping you to guide elements. You might also adjust the spacing of the grid (default is 8 pixels) inside the Options panel. Grid usage is a personal thing, and I find myself turning it on and off depending on how I feel at the time.

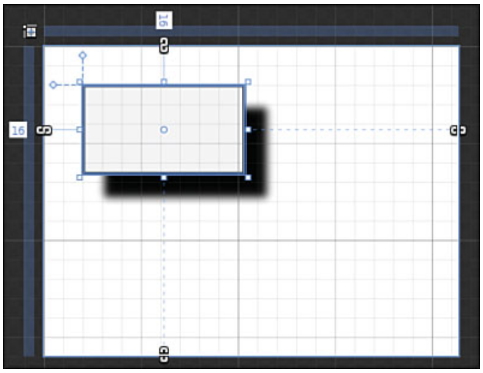


FIGURE 2.24 The default Grid effect applied to the artboard.

Snap to Gridlines
As you would most likely surmise, snapping to Grid enables you to move elements around the screen and have those element positions snap to your grid guides. Interesting to note is that you don't have to have the Grid shown to make elements continue to snap to grid.

Snap to Snaplines
Similar to smart guides in other applications, snaplines are approximations for boundaries that element share and other points of reference within your UI element collection. These snaplines can make it much more efficient to quickly place elements on the screen and get the alignment pretty close. Figure 2.25 shows two button elements that show the snaplines when one button is being moved close to it.

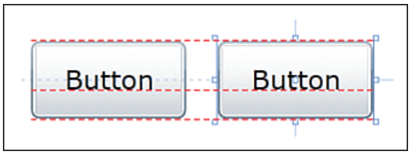


FIGURE 2.25 The snaplines alignment feature.

Show Annotations
Annotations are a feature used within Expression SketchFlow applications, enabling you to mark screens and components up with points of discussion visible to other users.

Figure 2.26 shows the creation of an annotation in against a SketchFlow button, whereas Figure 2.27 shows the same annotation collapsed, but remaining next to the element as the focus is shifted onto another element on the artboard.

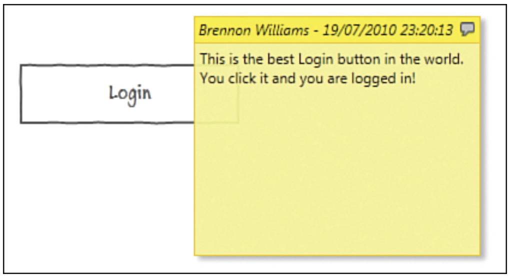

Login
Brennon Williams - 19/07/2010 23:20:13
This is the best Login button in the world.
You click it and you are logged in!

FIGURE 2.26 A new annotation is displayed.

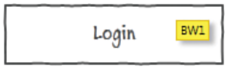
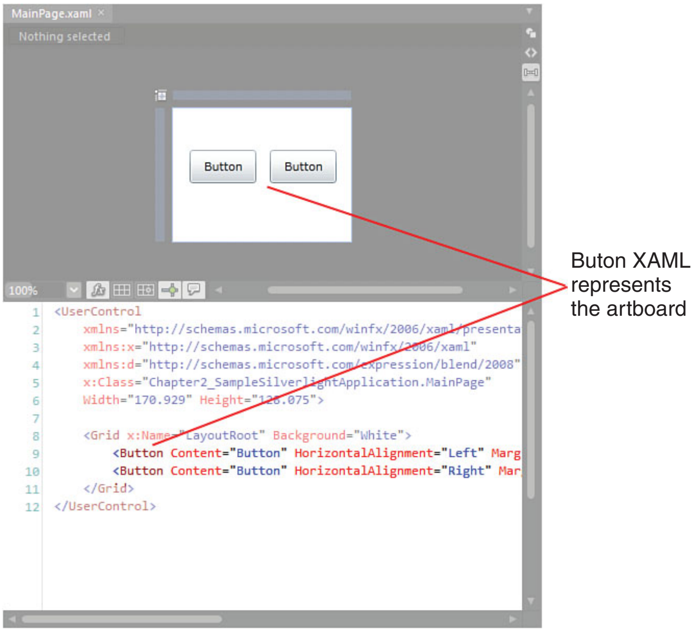


FIGURE 2.27 A collapsed annotation.

The XAML Editor

Figure 2.28 is showing a split view of both the artboard and the XAML window. In this case, it is the XAML markup that corresponds to the two buttons also shown on the artboard.

You can view just the XAML markup by itself if that is your preference, and you can also cycle between the design/markup/split view windows by clicking on F11. As shown in Figures 2.29 and 2.30, under the View menu, you have many choices within the submenus of the first two options.



```
<UserControl
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presenta
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    x:Class="Chapter2_SampleSilverlightApplication.MainPage"
    Width="170.929" Height="128.075">

    <Grid x:Name="LayoutRoot" Background="White">
        <Button Content="Button" HorizontalAlignment="Left" Marg
        <Button Content="Button" HorizontalAlignment="Right" Mar
    </Grid>
</UserControl>
```

FIGURE 2.28 The split view comprises both XAML markup and the artboard.

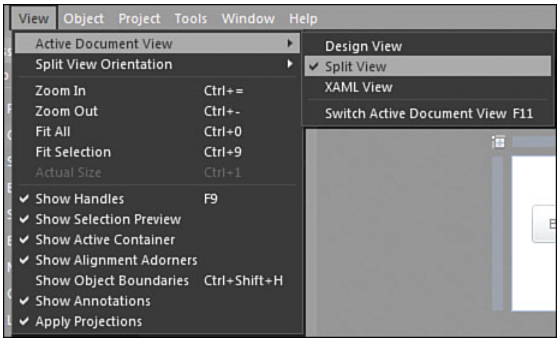


FIGURE 2.29 The Active Documents View options.

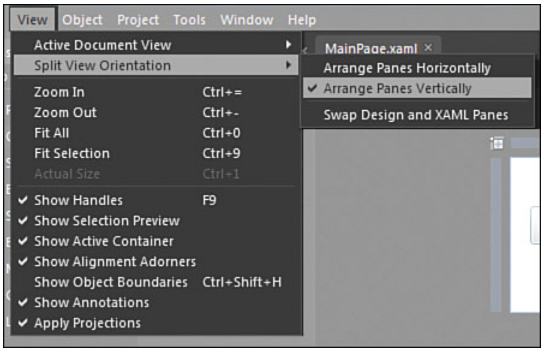


FIGURE 2.30 Split View Orientation options.

The first menu option, “Active Document View,” enables you to switch between the three split and solid view windows.

Split View Orientation options allow you to define exactly how you prefer to see the split in your windows. Horizontal or vertical, the choice is completely yours.

You should note that these options are not saved in custom workspaces, though.

The Code Editor

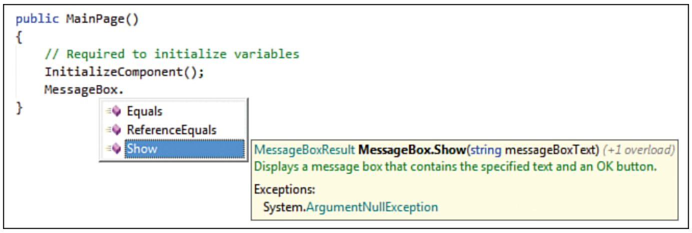
The code editor in Expression Blend has been added, removed, and then added again, much to a mixed reception. Some folks really want it to be in there, in case Visual Studio isn’t installed, or in some cases for simple editing of code where Visual Studio just isn’t required.

The art of writing code is very much out of the scope of this book, and there are many other texts that take you through the very beginning into the deepest of details around how code works and how you can write it, even with limited historical skills.

For coders, it does matter, though. The functionality of a coding editor and its ability to improve productivity are very much sticking points, and so it is no surprise that many people use Expression Blend 4 and Visual Studio 2010 together as an integrated toolset.

Intellisense Comparison

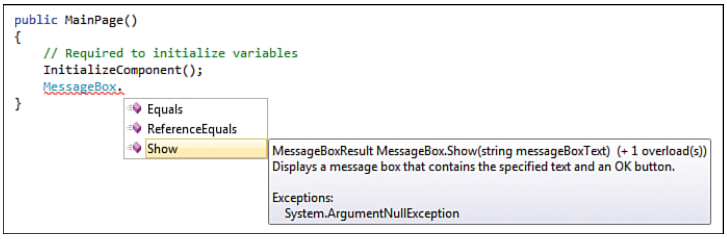
Figures 2.31 and 2.32 show how similar functionality in Intellisense is provided for in Blend versus Visual Studio, but as Figures 2.33 and 2.34 show, when you need to add library references fast and you can't remember the name of them, Visual Studio really shines through as the tool of choice.



```
public MainPage()
{
    // Required to initialize variables
    InitializeComponent();
    MessageBox.
}
```

MessageBoxResult **MessageBox.Show**(string messageBoxText) *(+1 overload)*
Displays a message box that contains the specified text and an OK button.
Exceptions:
System.ArgumentNullException

FIGURE 2.31 The Expression Blend 4 Code Editor Intellisense.

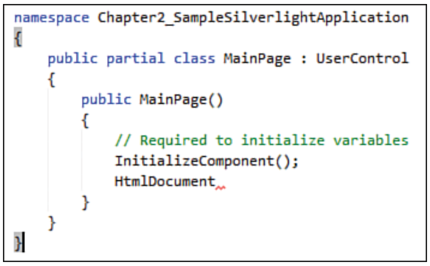


```
public MainPage()
{
    // Required to initialize variables
    InitializeComponent();
    MessageBox.
}
```

MessageBoxResult MessageBox.Show(string messageBoxText) (+ 1 overload(s))
Displays a message box that contains the specified text and an OK button.

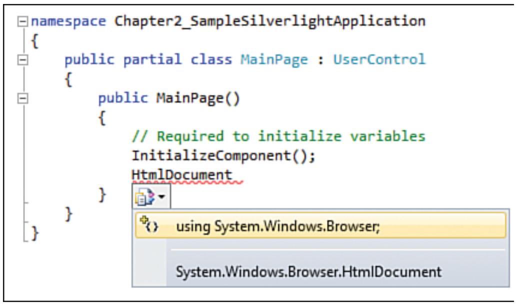
Exceptions:
System.ArgumentNullException

FIGURE 2.32 The Visual Studio 2010 Code Editor Intellisense.



```
namespace Chapter2_SampleSilverlightApplication
{
    public partial class MainPage : UserControl
    {
        public MainPage()
        {
            // Required to initialize variables
            InitializeComponent();
            HtmlDocument
        }
    }
}
```

FIGURE 2.33 Expression Blend shows nothing but a little red squiggle after the word "HtmlDocument," offering no help to find and use the correct code library.



```
namespace Chapter2_SampleSilverlightApplication
{
    public partial class MainPage : UserControl
    {
        public MainPage()
        {
            // Required to initialize variables
            InitializeComponent();
            HtmlDocument
        }
    }
}
```

using System.Windows.Browser;

System.Windows.Browser.HtmlDocument

FIGURE 2.34 Visual Studio gives single-click fast insert for the correct library.

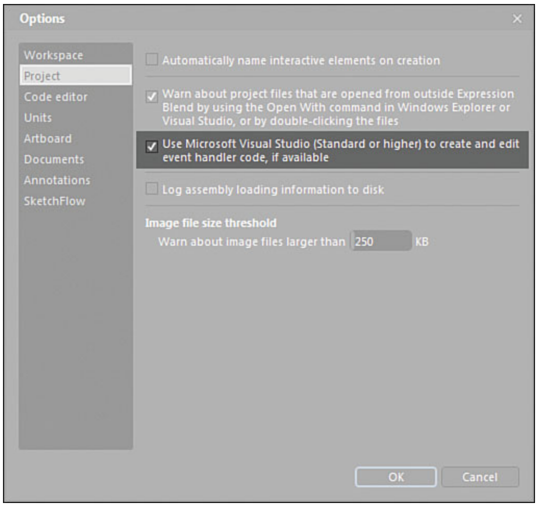
You might think that is just a little bit of more help, but with literally thousands of libraries that are available in the .Net Framework, trying to remember them all is next to impossible. Without the inclusion of those libraries (as indicated by the using statements in the code), you can quickly see how Visual Studio offers a much more robust coding experience.

Expression Blend and Visual Studio Integration

To enable Blend to work with Visual Studio seamlessly, you need to check a setting in the Tools->Options->Project dialog, as shown in Figure 2.35.

To try this out, the following steps show you how an event is handled between the two tools.

1. Create a new Silverlight Application + Website project.
2. Locate your Tool panel and the very last item ICON should be a Button element. Double click it to add a Button element to your screen.
3. Locate the Properties panel.
4. Name the Button element, as shown in Figure 2.36. Take note of the Event Viewer Mode button, detailed also in Figure 2.36, and click it after you have named your button.
5. Figure 2.37 shows you the large array of events that are available for a button. At this point, you should save your work by using the key combination: Ctrl + Shift + S at the same time or by selecting File->Save All.



☐ Automatically name interactive elements on creation

✓ Warn about project files that are opened from outside Expression Blend by using the Open With command in Windows Explorer or Visual Studio, or by double-clicking the files

✓ Use Microsoft Visual Studio (Standard or higher) to create and edit event handler code, if available

☐ Log assembly loading information to disk

Image file size threshold

Warn about image files larger than 250 KB

FIGURE 2.35 The setting to enable Visual Studio integration.

NOTE

Don't Worry If You Don't Understand the Code!

This is a simple exercise that is designed to show the integration between the Expression Blend and Visual Studio tools. Not understanding the code or the concept of event handlers is perfectly fine at this stage of the book.

NOTE

Properties Panel Overview

Shortly, you will be given a detailed tour of the Properties panel and how it works. For this example, you will have minimal interaction with it.

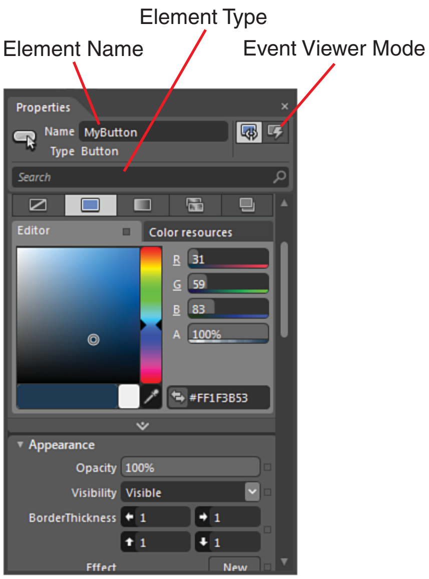


FIGURE 2.36 The Properties panel contains an area where you can give your elements a name as well as change the view of the panel to show all available Events.

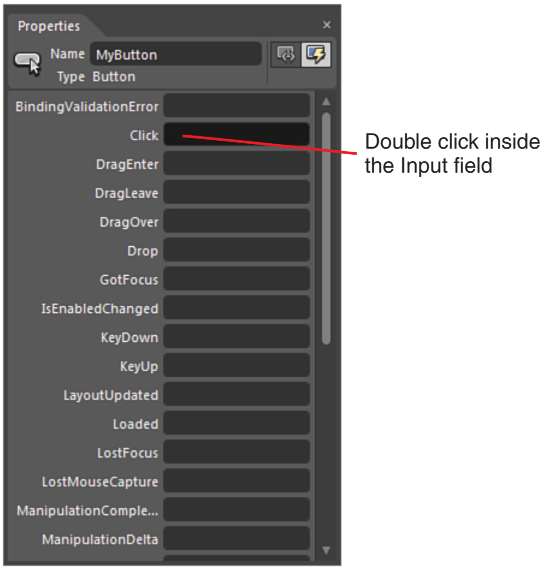
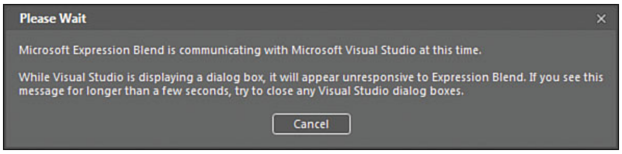


FIGURE 2.37 The Events view of the Properties panel.

6. Figure 2.37 also indicates where you should double-click in the event input field.
7. When you double-click in the "click" event input field, you might briefly get a message, as shown in Figure 2.38. This is always shown when Visual Studio is busy loading up.


Please Wait

Microsoft Expression Blend is communicating with Microsoft Visual Studio at this time.

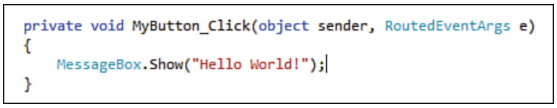
While Visual Studio is displaying a dialog box, it will appear unresponsive to Expression Blend. If you see this message for longer than a few seconds, try to close any Visual Studio dialog boxes.

Cancel

FIGURE 2.38 The alert that Expression Blend is trying to set up the communication between the two tools.

8. Visual Studio should have appeared after a quick load. If it hasn't, you might need to save all your work and open Visual Studio separately to ensure that it's working correctly. Also note that the Expression Blend 4 -> Visual Studio 2010 integration is only available for versions of Visual Studio that are standard or higher.

9. Figure 2.39 shows you the event handler code added by Visual Studio -> Blend interactions, as well as the code that you should write specifically into the generated handler inside Visual Studio.



```
private void MyButton_Click(object sender, RoutedEventArgs e)
{
    MessageBox.Show("Hello World!");
}
```

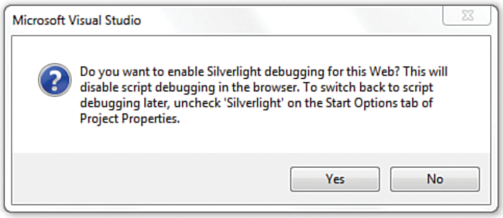
FIGURE 2.39 The newly generated event handler and the code to add inside the handler method.

10. Run the application from within Visual Studio by pressing the F5 button, which is the same as in Blend.

Setting Debug Mode
What you should get the first time you run the application from within Visual Studio is the dialog shown in Figure 2.40. This dialog is asking about adding some debug settings to the solution, which you should add now if you know that you will need to debug your application either now or at a later stage.

Figure 2.41 shows another dialog asking for the specific location of the debug file that must be generated and added to your project. You can simply opt OK at this stage, where a Web.Config file will be added to your project solutions.

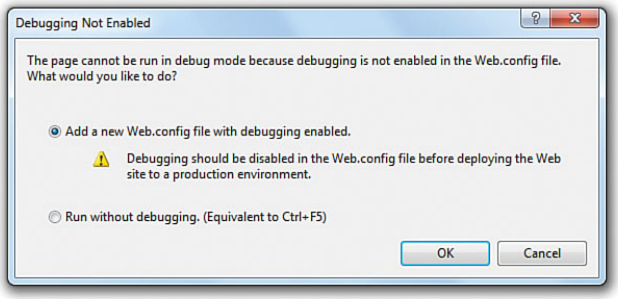
Your application should now start up and run inside your default browser, and if you have installed Silverlight correctly, you should see your button showing on the screen where you can now click on the button, and a MessageBox should be presented with "Hello World" contained, as shown in Figure 2.42.



Microsoft Visual Studio

Do you want to enable Silverlight debugging for this Web? This will disable script debugging in the browser. To switch back to script debugging later, uncheck 'Silverlight' on the Start Options tab of Project Properties.

FIGURE 2.40 The dialog asking about Debug mode from within Visual Studio.



Debugging Not Enabled

The page cannot be run in debug mode because debugging is not enabled in the Web.config file. What would you like to do?

[✓ Add a new Web.config file with debugging enabled.]

Debugging should be disabled in the Web.config file before deploying the Web site to a production environment.

Run without debugging. (Equivalent to Ctrl+F5)

FIGURE 2.41 How Visual Studio asks you where it should generate the Debug file.

You can do it over and over, but that will get tiring very soon.

11. Close the web browser for now and then return to Visual Studio. Save your work by pressing the Ctrl + Shift + S combination and then return to Expression Blend.

12. Locate the Project tab and, as shown in Figure 2.43, locate the ".cs" file that represents the code-behind file of your MainPage.xaml. By double-clicking on this file, you will again see the code editor in Blend, and it should now show the updated code that you added to the event handler in Visual Studio.

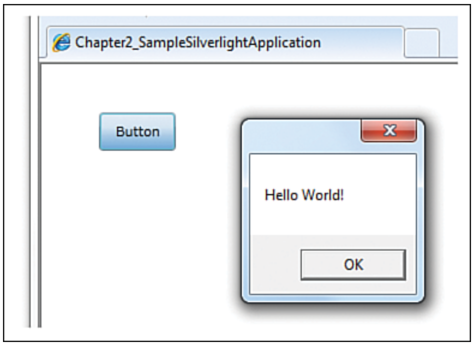


FIGURE 2.42 The result of the button click calling the event you added in code and raising the MessageBox.

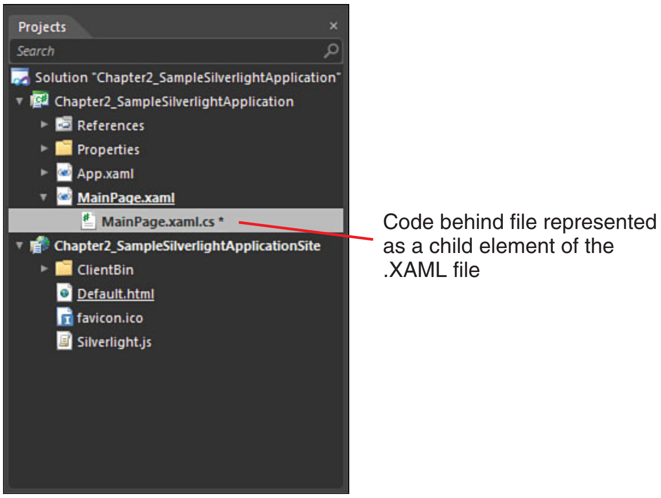

Projects
Search
Solution "Chapter2_SampleSilverlightApplication"
Chapter2_SampleSilverlightApplication
References
Properties
App.xaml
MainPage.xaml
MainPage.xaml.cs *
Chapter2_SampleSilverlightApplicationSite
ClientBin
Default.html
favicon.ico
Silverlight.js
Code behind file represented as a child element of the .XAML file

FIGURE 2.43 The location of the code-behind file in the Project tab.

Simple Edits Inside of Blend

You can now inside the Blend code editor, change the "Hello World" string to anything else you want, and run the application with F5 directly from Blend. This simple example demonstrates that it's easy to modify and work with code inside of Expression Blend, but anyone thinking they would be easily be able to code up a fully-fledged application in it would be met with frustration very quickly.

Objects and Timeline Panel

The Objects and Timeline panel represents hierarchical views of all the elements that are currently in the scene you are working in. You can translate that simply to mean that it shows parent-child relationships.

The Objects and Timeline panel also provides a visual representation for attached objects, such as Behaviors (see Chapter 3 for more details), that don't necessarily have a visual representation on the screen but are indeed attached to a specific element.

As the name of the panel also suggests, it contains the primary Timeline manager for the Blend application, which you cover in depth in Chapter 11, "Animations and Transitions."

Figure 2.44 details the most important parts of the Objects and Timeline panel, and it is very important that you become familiar with it as quickly as possible because this will be one of the main panels you interact with when using Expression Blend.

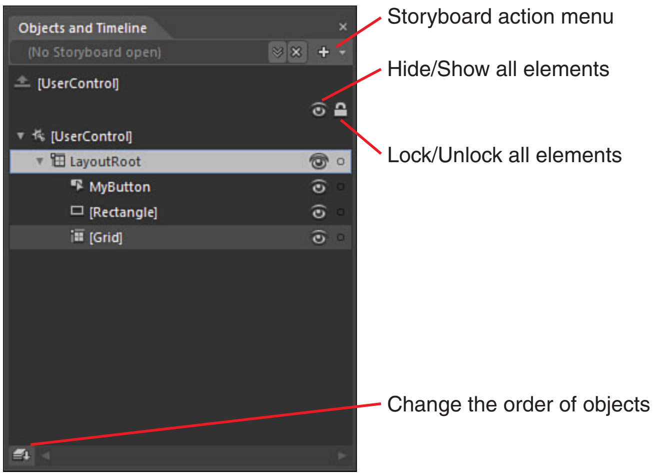


FIGURE 2.44 The Object and Timeline panel details specifically for objects.

Locking Elements

Figure 2.44 also points out that you can control the locking state of individual objects or elements and their children simply by clicking on the Lock icon. When locking is engaged, you are restricted from selecting the item on both the artboard and in the Objects and Timeline panel.

Dragging Elements

The collection on display is simple to move around in terms of Z-Order. Z-Order refers to how elements sit on top of one another on the screen. You can simply drag elements up and down the list shown to change their parent relationship or how they are presented on the screen. You can actively drag items from the Assets panel and toolbar onto the Objects and Timeline panel, as well as various Behaviors and effects, as shown in Figure 2.45.

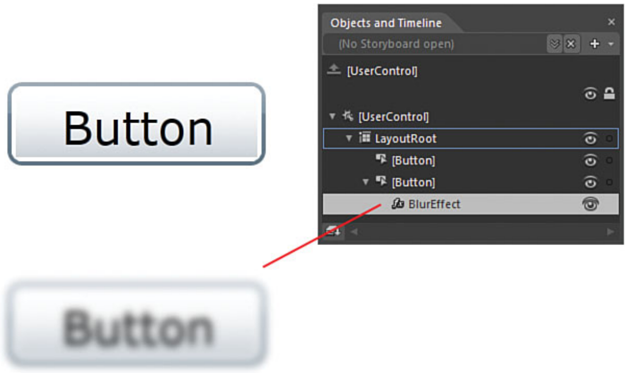


FIGURE 2.45 A Blur effect applied to a Button element.

Even in the scenario shown in Figure 2.45, you can simply drag the Effect element from one Button element to another.

The Objects and Timeline panel contains a multitude of functionality that is exposed even further when working with storyboard animations. Figure 2.46 shows how the panel expands and contains a fully-fledged Timeline editor. See Chapter 11 for further details.

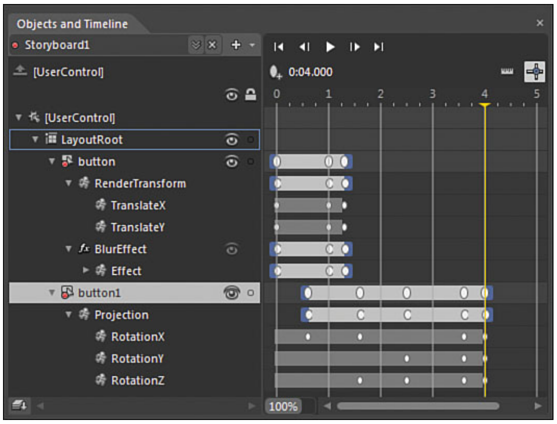


FIGURE 2.46 The Object and Timeline panel in full flight with storyboard editing.

Tools Panel

The Blend Tools panel hasn't seen much evolution since version 1, with the exception of being dockable and the modification to some of the Gradient tools that are available from it.

As Figure 2.47 demonstrates, the Tools panel changes slightly depending on the application type: WPF or Silverlight.

The Tools panel is made up of several collections of controls, including primary artboard tools such as Pan, collections including layout controls, shapes, text controls, and common controls.

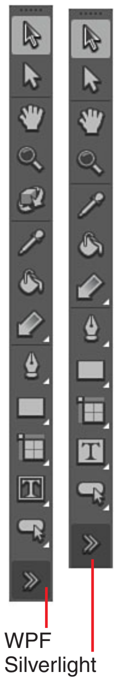


FIGURE 2.47 Displayed is the slight variation in WPF and Silverlight Tools panel.

Selection Tools

The Selection tools contain two conceptually similar tools that used within different context of editing.

Selection Tool

The Selection tool enables you to grab elements and move them around the artboard.

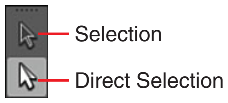


FIGURE 2.48 The two Selection tools available.

Direct Selection Tool

The Direct Selection tool, as shown in Figure 2.48, is used for selecting path segments.

View Tools

The View tools collection contains three tools used to control your view of the design surface.

Pan Tool

The Pan tool enables you to literally grab the entire artboard and move it around your screen. The shortcut key to allow you to do this at any time is to hold down the spacebar and move your mouse to the location of your choice.

Zoom Tool

The Zoom tool enables you to accurately click on an point of the artboard and zoom in or out. To enable zoom out, you simply hold down the Alt key and continue clicking your mouse until you reach the desired zoom level.

Camera Orbit Tool

This tool is only available when you are authoring a WPF application and is quite cool to play around with.

The following steps take you through creating a fast and simple 3D element to help you experience the Camera Orbit Tool.

1. Create a new WPF application.
2. In the Project tab, right-click on the project name element and select Add Existing Item, as shown in Figure 2.49.
3. You should be provided with an "Add Existing Item" dialog box where you can navigate your files. I have chosen an image of tulips, which is part of the Windows 7 Sample Pictures file, but you can choose whatever image you want to use.
4. If you have chosen a moderately large image, you see the dialog showing in Figure 2.50. This is an alert from Blend that tries to make you conform to the good practice of packaging your large image files with your EXE files. I also recommend you accept this advice and select Yes to continue.
5. If all has gone as expected, you should now be able to see your image inside the Project panel and being an image, if you role your mouse over it, you should see a small Thumbnail image, as shown in Figure 2.51. All you need to do now is drag that image file directly from the Project panel to the artboard, or you could right-click on the image and select "Insert" from the context menu that appears.

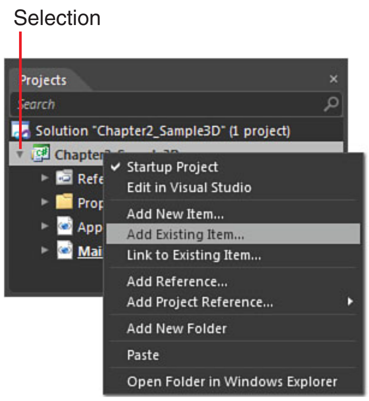
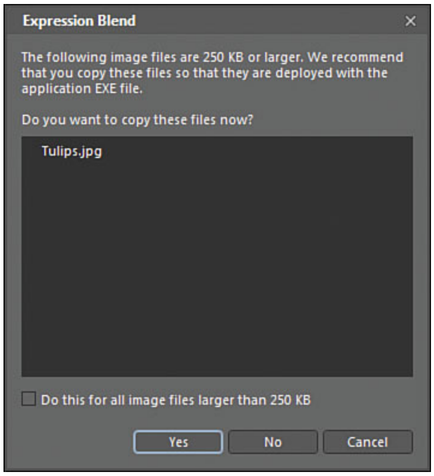


FIGURE 2.49 Add an existing item to your project.


Expression Blend

The following image files are 250 KB or larger. We recommend that you copy these files so that they are deployed with the application EXE file.

Do you want to copy these files now?

Tulips.jpg

☐ Do this for all image files larger than 250 KB

FIGURE 2.50 Showing the large file alert provided by Blend.

6. Most likely, the image will be massive and might indeed be larger than your artboard. Use the Selection tool to resize the image so you can work with it in the center of the artboard.

7. Take a quick look at the Objects and Timeline panel, as shown in Figure 2.52, and note that currently you should simply have an Image element as the child control of the LayoutRoot element.

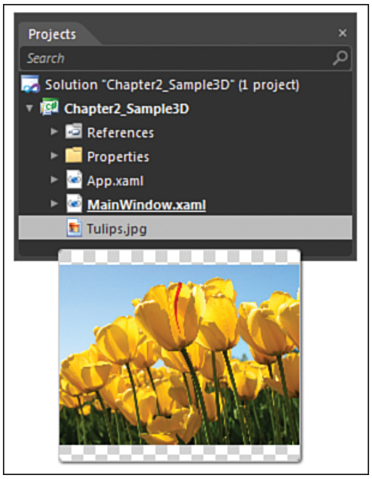


FIGURE 2.51 Image resources are presented in the helpful thumbnail provided by Blend in the Project panel.

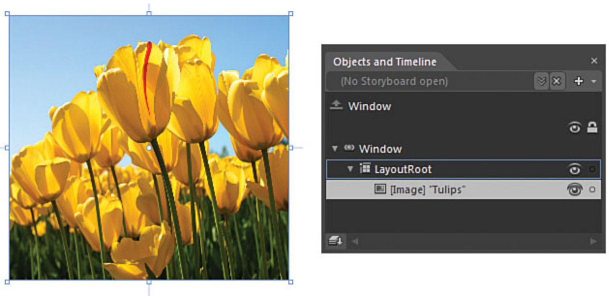


FIGURE 2.52 The Image element is shown as a child element of the LayoutRoot element.

8. With the Image element selected, click on the Tools menu and then select the "Make Image 3D" menu item.

9. Nothing will look like it has changed, but if you now take another look at the Objects and Timeline panel, you see that the Image tag has gone, and in its place is a Viewport3D element, as shown in Figure 2.53.

10. There is a lot you can do from here, but at the moment, you are looking at the Camera Orbit tool in the Tools panel. Select the Camera Orbit tool, click on the image with your mouse, and keep your mouse button down, dragging across the image.

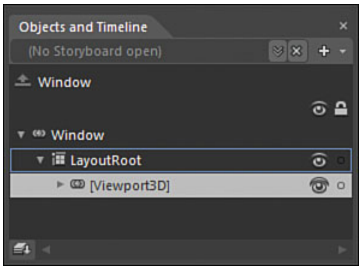


FIGURE 2.53 The Objects and Timeline panel change.

11. You should see your image appearance change, which is indeed the 3D Camera position changing.

    To view some more accurate details around this Camera Orbit, you can drill down through the Viewport3D element in the Objects and Timeline panel, finding and then selecting the PerspectiveCamera element, as shown in Figure 2.55.

12. Open the Properties panel, which you should now see has some very specific properties for the camera, as shown in Figure 2.56. Move the Camera Orbit tool back over the image on your artboard and keep an eye on the Position, Direction, and Up Vector properties specifically, and you should see them changing to actions you are performing on the artboard.

Experiment with those tools and property values inside the Camera properties, changing camera and dragging on the interactive property setters, as also referred to in Figure 2.56.

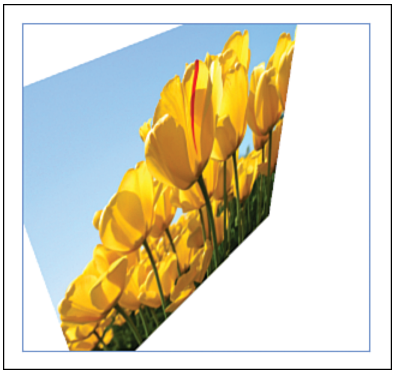


FIGURE 2.54 Showing the result of Camera Orbit tool used against the 3D element.

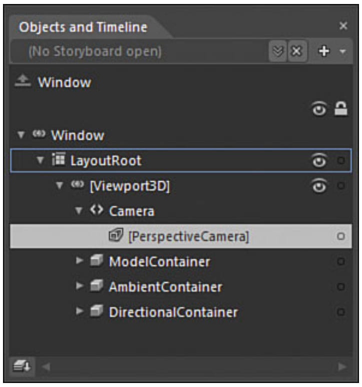
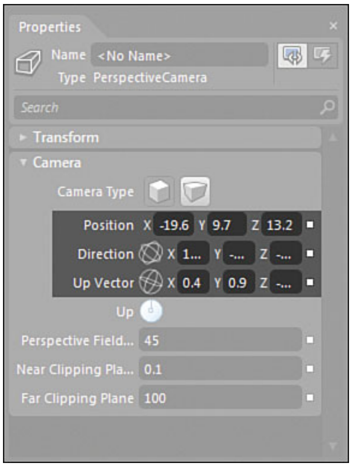


FIGURE 2.55 The PerspectiveCamera element in the Objects and Timeline panel.



Properties
Name <No Name>
Type PerspectiveCamera
Search
Transform
Camera
Camera Type
Position X -19.6 Y 9.7 Z 13.2
Direction X 1... Y -... Z -...
Up Vector X 0.4 Y 0.9 Z -...
Up
Perspective Field... 45
Near Clipping Pla... 0.1
Far Clipping Plane 100

FIGURE 2.56 The camera-specific properties affected by the Camera Orbit tool.

Brush Tools

The first tool I believe is incorrectly named or incorrectly propositioned as an eyedropper control. You would naturally associate an eyedropper with a color selection tool, which it is in the case of Blend; what this particular eyedropper tool does, however, is duplicate some matching properties of one element to a matching property collection of another element, not just color.

It sounds confusing, so follow through with this example to understand the usage of the tool:

1. Create a new Silverlight Application + Website project.
2. Add two Button elements, a rectangle and a Border control, so your layout is the same as shown in Figure 2.57. Take note of the content of the buttons, identifying them as Button 1 and Button 2.

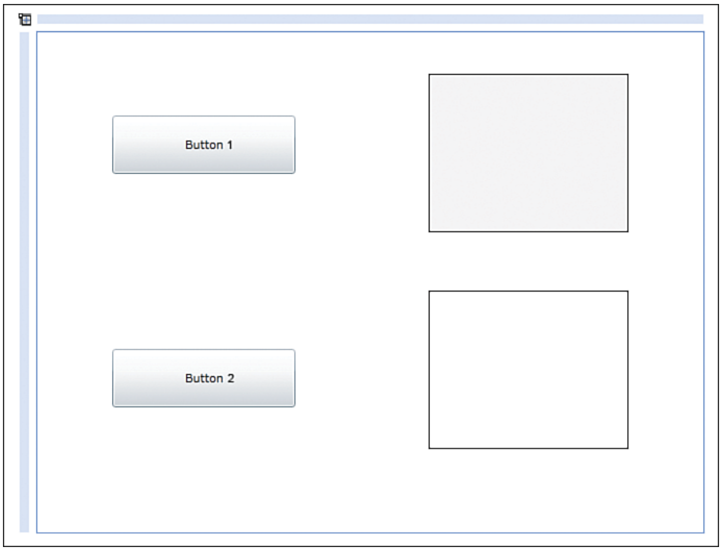


FIGURE 2.57 Setting up with the sample layout.

3. Select Button 1, and note specifically in the Properties panel that the Button element has a background, BorderBrush, and a foreground color property.
4. With Button 1 still selected, change the Background property to a solid color bright green, the foreground to a bright pink/purple and the BorderBrush property to a solid color bright red. You end up with something like the image shown in Figure 2.58. Note that I have increased the BorderThickness property of both the buttons to 4 pixels all

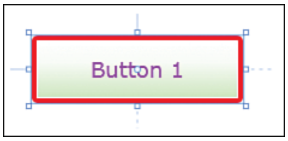


FIGURE 2.58 The setup properties.

around and increased the font size to 18 pixels to accentuate the effect.

5. Select Button 2.
6. Select the Eyedropper tool from the Tools panel and click once on Button 1. You should notice that Button 2 instantly takes across the entire set of matching color properties, as well as the font size from Button 1 to Button 2.

NOTE

Why Is the Button Background Not a Solid Color?

This is entirely expected and is because of the default template applied to the Button element. For now, don't worry too much about this. Chapter 6, "Element Styles and Templates," walks you through this entire process.

7. If you now select the Rectangle element (which is the top right) and again select the eyedropper, click again on a button with the Eyedropper tool. You see that nothing happens at all. This is because the properties don't match for a rectangle and a Button element. The rectangle has a Fill property instead of a Background property.
8. Select the Border element and again select the eyedropper and select a button. Did you get what you expected? The background of the border is the solid color of the Buttons Background property, and the BorderBrush property is duplicated as well, but as you can see, the BorderThickness is not copied, as shown in Figure 2.59.

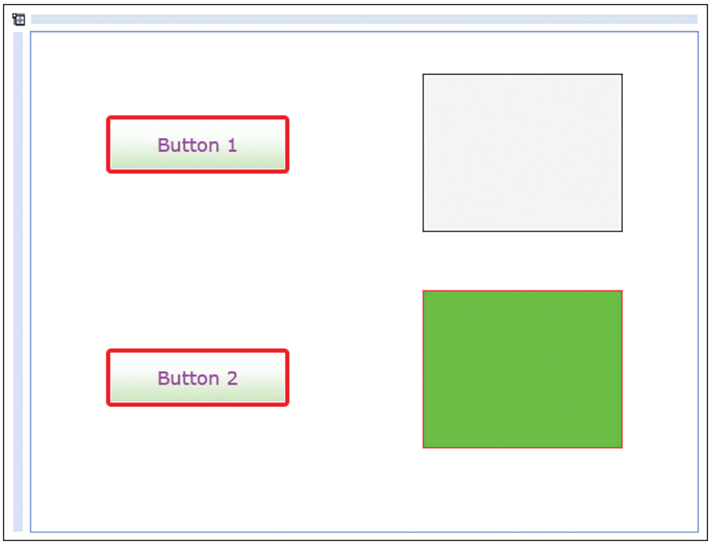


FIGURE 2.59 The result of the Eyedropper tool.

It's confusing, and I have made these points to the Expression Blend team. You might find some use for it—I can't.

The Paint Bucket tool is the same sort of confusion, but in reverse. To try this out, the following sample quickly shows the effect:

1. Change the layout of your app slightly and add another Border element, as shown in Figure 2.60. I gave the new Border element a thicker border (4 pixels), and you can apply similar colors.

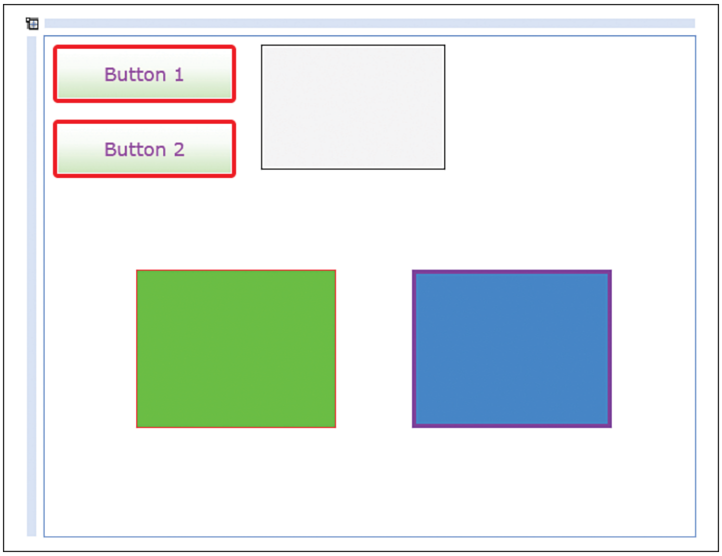


FIGURE 2.60 The new layout of the sample application.

2. With the new Blue Border element selected, select the Paint Bucket tool and then click on the existing green Border element.

I am sure you noticed that the BorderThickness property was not translated.

The next two tools are true Brush modifiers, the Gradient tool and the Brush Transform tool, and are shown in the collection detailed in Figure 2.61.

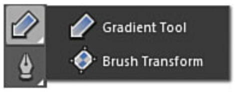


FIGURE 2.61 The Brush modifier tools collection.

You now use these two tools to understand each one in isolation; later in Chapter 4, “Common Properties and Functionality,” you see how these two tools can work even further to adjust and help you work with patterns and image brushes.

The following steps walk you through creating and modifying a GradientBrush created with the Gradient tool.

1. Modify the layout of your solution so the Rectangle element takes the most space, as shown in Figure 2.62.

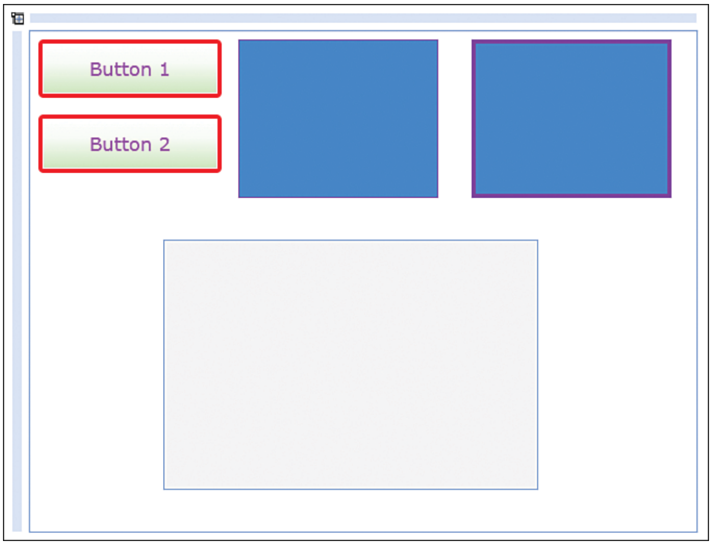


FIGURE 2.62 The new sample layout.

2. Select the rectangle and then the Gradient tool (the one with the Arrow icon).
3. Place the mouse cursor slightly inside the top-left corner of the Rectangle element. With the left mouse button held down, drag your mouse to the bottom right of the rectangle.
4. You should end up with a weird-looking gradient, black and white, stretching across the rectangle. Notice in Figure 2.63 the Gradient Stop modifiers. You can now select either one of those little rings and move them up and down the gradient guide line to change the weight of either color.

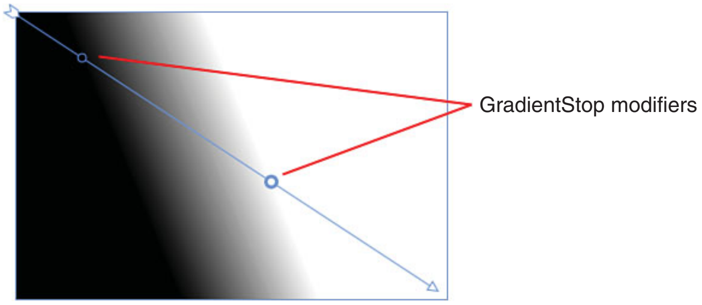


FIGURE 2.63 The Gradient Tool guide.

5. You can move the guide directly with the mouse, or you can move the mouse to either end of the guide (the arrow tip or end), and you see that you can shorten or lengthen the guide. You can also rotate the guide by moving the mouse slightly past the end of the guide.

6. Hold down the Shift key while rotating the guide, and you see that the guide turns in locked 15-degree increments.
7. Select the Brush Transform tool.
8. You see that the guide system changes considerably to present you with a rectangle type shape, as shown in Figure 2.64. This guide is easy to move around and independently change the depth and width of the guide, which gives you more accuracy when working with specific gradients.

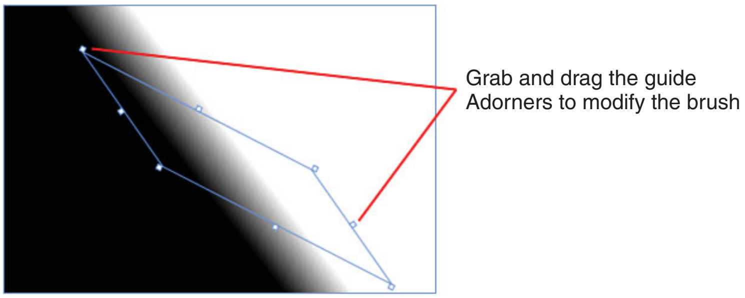


FIGURE 2.64 The Brush Transform tool guide system.

Object Tools

The Object tools are multiple collections that contain a mixture of common controls, Layout panels, and Path and Shape tools. Most of these tools are covered in depth throughout the rest of the book, so to save on confusion at this point, I will detail the collection in the following and point you to the chapters that contain more details.

Path Creation Tools

The Pen tool, as shown in Figure 2.65, is used to create fixed-line path elements, connected from point to point. The tool has multiple key modifiers to help you create very accurate arcs and tangents.

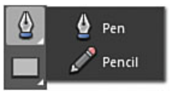


FIGURE 2.65 The Path creation tools.

The pencil represents a freehand drawing tool, enabling you to plot path line elements everywhere that you hold the mouse down on the artboard.

See Chapter 12, “Shapes, Paths, and Effects,” for details on using the Path creation tools.

Shape Tools

As you would expect, the Shape tools enable you to create and modify simple shapes, such as the rectangle, ellipse, and line, as shown in Figure 2.66.

See Chapter 12 for more detail on using the Path creation tools.

Layout Panel Collection

As shown in Figure 2.67, the Layout panel collection contains several panels that specifically control the layout of controls that you use in your user interface. The primary Layout panel is the grid, and it is very important that you understand how to use this control property in order to master your use of Expression Blend.

Also note that in Figure 2.67, WPF type projects have more tools available, which is because of the larger control set that WPF contains.

See Chapter 5, "The Art of Layout," for greater detail on how to correctly use layout.

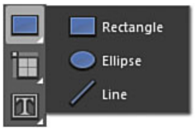


FIGURE 2.66 The Shape tools.

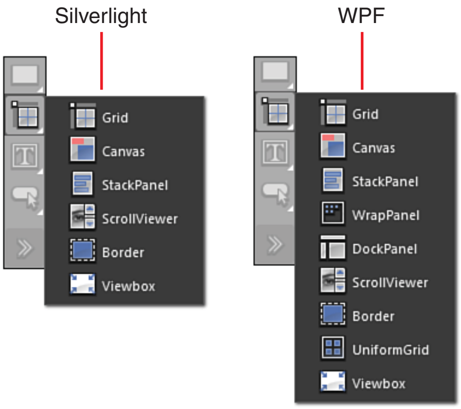


FIGURE 2.67 The Layout panel collection for both Silverlight and WPF.

Text and Input Controls Collection

Some Text input controls have specifically defined functionality such as the PasswordBox, which enables you to help conceal sensitive information. The TextBox is an input control, whereas the TextBlock and Label controls have specific usage around the display of text back to your user.

As shown in Figure 2.68, both Silverlight and WPF share some common controls such as the TextBox, PasswordBox, and RichTextBox, for example.

See Chapter 4, "Common Properties and Functionality," for more details around using Text input controls.

NOTE

Why Does WPF Have More Controls Than Silverlight?

Silverlight is a subset of Windows Presentation Foundation (WPF), and in order to reduce the install size of the Silverlight solution, many classes and controls were removed from the .Net Framework version that supports Silverlight. Most of the removed controls are available from the Silverlight SDK or additional libraries.

Common Controls

The most commonly used controls such as the Button and ListBox are always in easy reach for both Silverlight and WPF by being part of the common controls collection, as shown in Figure 6.69.

See Chapter 4 for more details around using Text input controls.

Asset Tools

The Asset library tool button gives access to the Assets panel, which will be detailed in the next section.

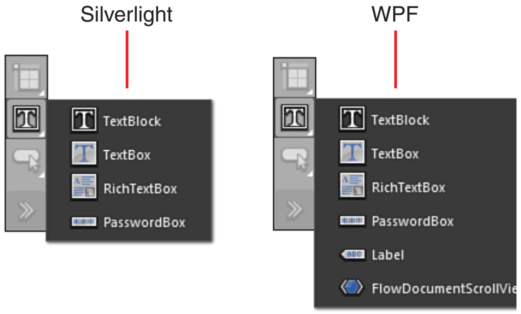


FIGURE 2.68 Some of the common controls shared by Silverlight and WPF.

Assets Panel

The Assets panel represents collections of all available controls, media, effects, Behaviors, and much more, as shown in Figure 2.70.

What Figure 2.70 doesn't indicate is that a WPF projects contains the same layout and functionality of the Assets panel, but different controls that are not available in Silverlight (this is the same in reverse) and WPF projects typically contain many more reference locations.

Usage is pretty self-explanatory, in that you find a control or behavior, for example, that you want to use, and you either double-click the item or drag directly to the artboard or the Objects and Timeline panel.

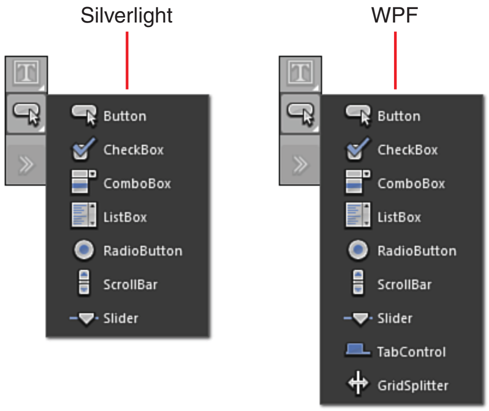


FIGURE 2.69 The Common Controls collection for both Silverlight and WPF.

Searching

As you become more familiar with various controls and elements, you will naturally start to remember the names of the controls and elements, so the Search mechanism of this panel is very helpful. You only need to begin typing the name of what you are after and the collection will begin to modify to show you what is available, as shown in Figure 2.71.

As I typed in the letters "he," you can see in Figure 2.71 how the collection has now reduced considerably, and you see that the collection count indicators also show relevance to the available result.

Properties Panel

The Properties panel contains a substantial amount of functionality and represents (along with the Objects and Timeline panel) one of the primary areas that you will spend most of your time in when working in Expression Blend, so learning how to navigate it is very important.

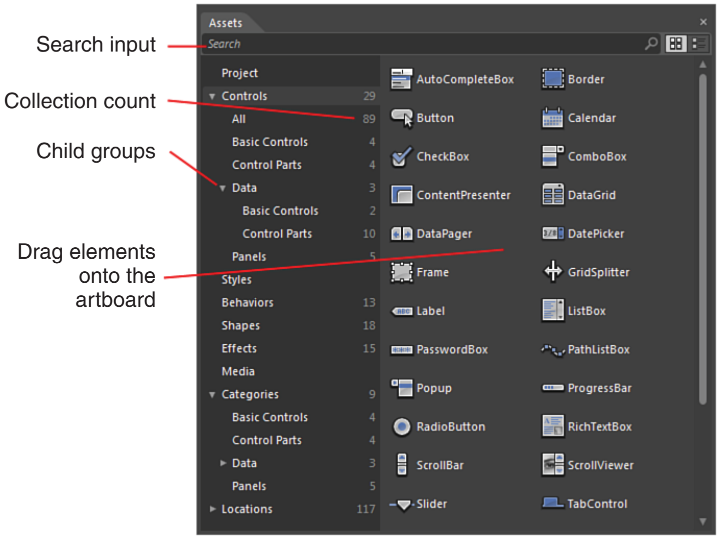
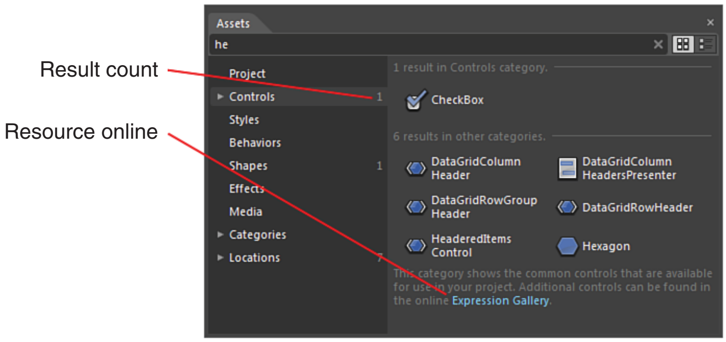


FIGURE 2.70 The Assets panel.



This category shows the common controls that are available for use in your project. Additional controls can be found in the online Expression Gallery.

FIGURE 2.71 The Asset panel context change in the collection after searching.

The following section details the most-used features of the Properties panel, although it should be noted that the panel and the collections shown inside the Properties panel are context driven, and will change dramatically depending on what you are doing with Blend at the time.

Figure 2.72 indicates that the Properties panel also has an Event Viewer mode that will be detailed last in this section.

NOTE

Changing Controls Changes the Category Collection

In this section, the Properties panel is detailed while a Button element is selected on the artboard, as this will show the most common functions and features of the panel.

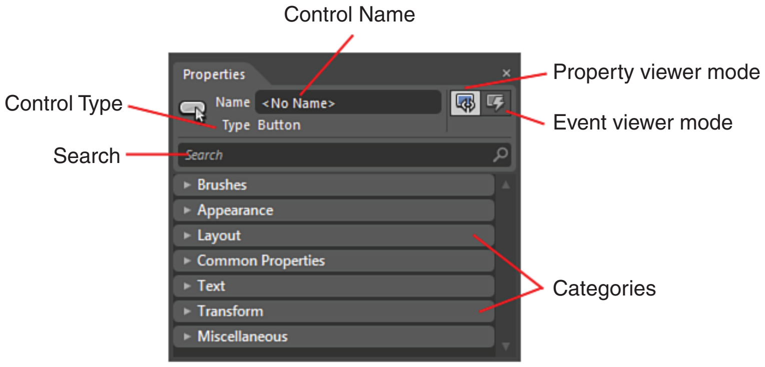


FIGURE 2.72 Overview of the Properties panel.

You see that several Property categories are very similar in their layout and contain simple input boxes that you can type values into directly, or use your mouse to drag values up and down in the case of numbers.

Name and Type

The name of any element is by default <No Name>, which means that the control or element has no Name value assigned to it. Depending on the architecture of your application, you might choose to use names and if so, it is advised you take on a constant naming convention for use in your applications. By default as mentioned, Blend does not assign a name to your controls, but this rule is broken when you apply behaviors or specific functionality to a control, such as animate a property of a button, in which case Expression Blend assigns a default name to your element. This is done so your application can represent the correct instance of your control elements. Think about it for a minute, if you had 10 buttons on your artboard and only wanted to animate one of those buttons, the application needs to be able to reference that specific control.

The *Type* of control is very specific and is well worth noting when you add or work with new controls. You can clearly see in Figure 2.72 that Button element is currently being reviewed and the Property collection also shown in Figure 2.72 is the property collection representation of the Button type control.

NOTE

Property Changes Occur Implicitly

What does this really mean? Expression Blend will change the property value of your elements automatically when you work with them on the artboard, so for instance, if you select a Button element on the artboard and resize it, you see that the appropriate Width properties will also change. See Chapter 5 for more details on how properties are also affected by Parent elements.

Search

Search is a powerful feature of the Properties panel and can help significantly reduce confusion and aid in productivity. Figure 2.73 shows how the Category collection has been reduced, and only properties that contain a pattern match of "cont" are shown.

My advice is to use property searching as often as possible.

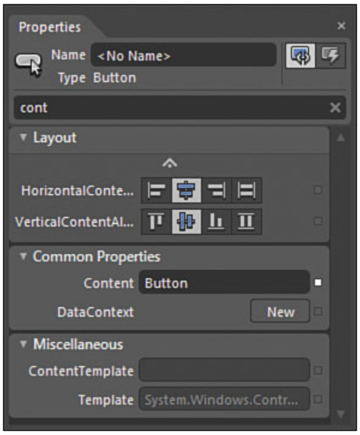


FIGURE 2.73 The result of searching the Property collection.

Categories and Advanced Properties

As you have seen in varying figures, there are plenty of categories that properties are divided into. What you also see in the Properties panel is that some categories contain an Advanced Property section that extends the property collection, sometimes with only a single property and sometimes much more, as shown in Figure 2.74.

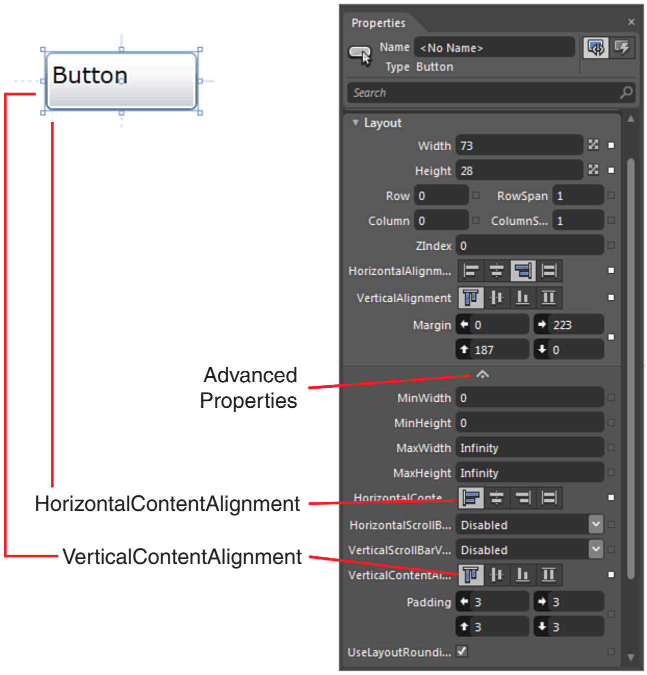


FIGURE 2.74 The Layout category and Advanced Properties for layout.

Figure 2.74 also shows you the effect that the Advanced Properties of HorizontalContentAlignment and VerticalContentAlignment have on the Button element also shown.

Property Binding References

Ignore the fact that the control type in Figure 2.75 is a ContentPresenter; I wanted to show you here that Blend indicates if properties are not their default values, as well as indicating that the value supplied to the property is coming from somewhere else.

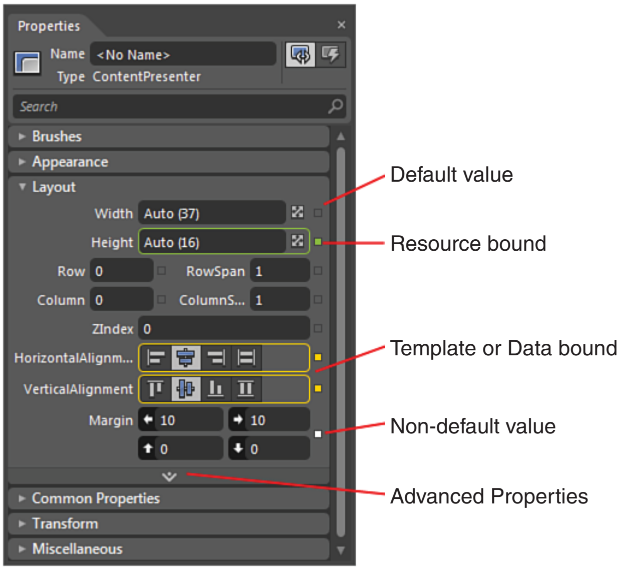


FIGURE 2.75 Different property binding notifications.

In the case of the green border and box, this is indicating that the value supplied is coming from a resource within the application. A resource is a value that is usually fixed—it could be a template, or it could be a number, as in this case. See Chapter 13, “Skins, Themes, and Resource Dictionaries,” for more details on resources.

If the border and box are orange, this indicates that the property value is “bound” to a property somewhere else in the application. This is a complex concept that is out of scope for this section of this chapter, but reviewing Chapter 6, “Element Styles and Templates,” gives you the detailed understanding of what this all means.

A white box means that the value applied is not the default value of that particular property and, conversely, an empty or clear box means that a default value is applied.

Advanced Options

The Advanced Options dialog, as shown in Figure 2.76, is a combination of extended functionality and shortcuts. The options as they are shown become enabled/disabled when working in specific context.

An example of this is that Figure 2.76 is shown as the result of clicking on the green box for the property that is currently bound to a resource; you will see that the options to "Edit Resource…" and "Convert to Local Value" are enabled.

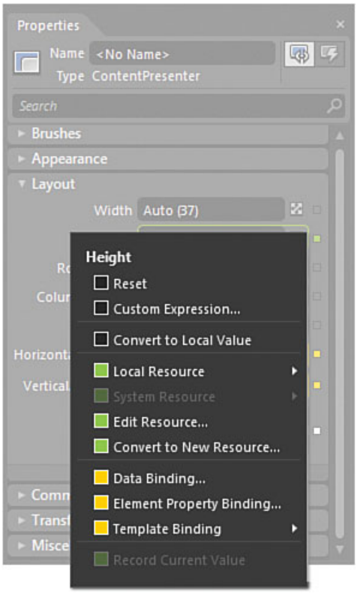


FIGURE 2.76 The Advanced Options context menu.

For the most part, these options are explained in context throughout the rest of the book and will not be detailed here. The last advanced option, "Record Current Value," is only available when working within a storyboard for instance, so its usage will be shown in Chapter 11, "Animations and Transitions."

Brushes

The Brushes category, as shown in Figure 2.77, carries significant functionality that takes a little practice to really come to terms with. Understanding where everything resides is half the battle, but you should quickly become comfortable with working in the color pallets for SolidColorBrush and GradientBrush types. Case in point is locating where you can switch from RGB to CMYK color values, which is hidden away completely. Figure 2.78 shows that you can click on any one of the letters of RGB, and you will be presented with a little context menu offering your choice for saturation, hue, and other modifiers.

Resources play a large part in how you will interact with brushes and the properties that they are applied to and, as such, will be detailed in Chapter 13, "Skins, Themes, and Resource Dictionaries" As a matter of practice, you will use the Brushes category heavily throughout the rest of the book.

Active Icons

Some specific categories such as brushes have their own unique property editing experience, but it's interesting to note as well that some categories have what appear to be just icons but are, in fact, interactive property assistants.

One of the most common property categories is called Transform. You will learn more about Transform in Chapter 11, but for now, it serves as a great example of active icon assistants in Blend.

If you create a Silverlight Application + Website project and add a simple button to the artboard, you should see that the Transform category is available in the Properties panel.

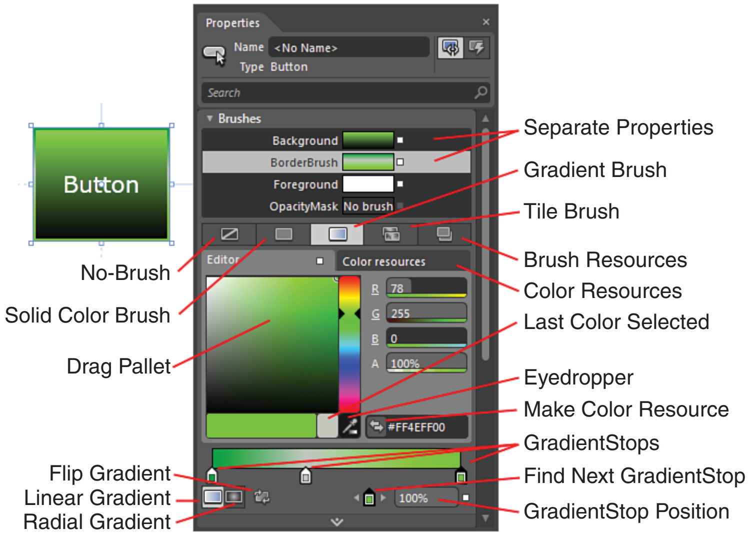


FIGURE 2.77 A high-level overview of the Brushes category.

Figure 2.79 is showing the effects of a Plane Projection Transform applied to the button element, which in this case is rotating the element in all three axis. What I am pointing out here, though, is that I didn't need to enter the values into the X, Y, and Z input boxes. You can simply drag your mouse of the active icon on the left.

Most Transform properties have an active icon, so have play around with the effects that they create.

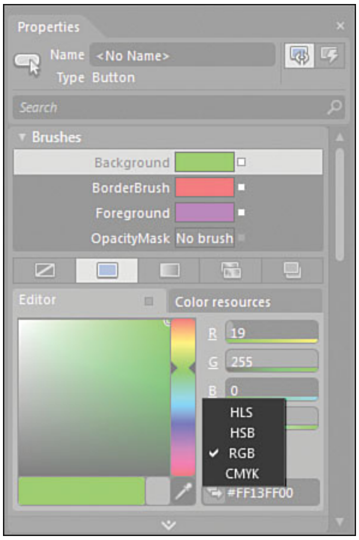
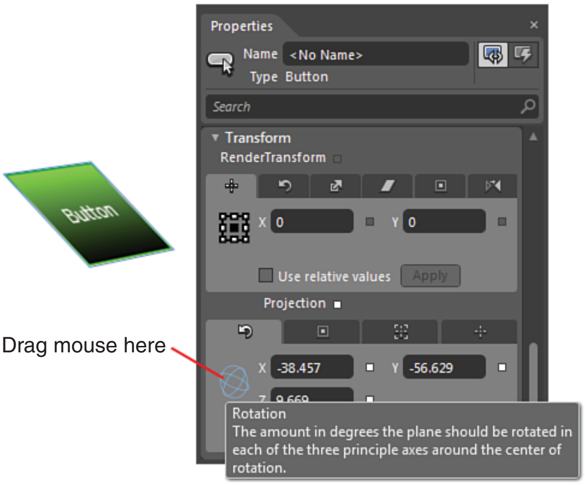


FIGURE 2.78 The very well-hidden options to switch color modifiers.

Events

Events occur when a user or dynamic parts of your application cause something to happen to an object. The easiest way to understand this is to consider a Button element. When you physically click on a button, you are causing an event to occur—the Click event, to be specific. Events enable you to react to the Click and perform whatever task suits your requirements. For instance, a click could mean that someone has entered a value on a form, and the button is a Submit button, and so on.



X 0 Y 0

X -38.457 Y -56.629

Rotation
The amount in degrees the plane should be rotated in each of the three principle axes around the center of rotation.

FIGURE 2.79 Demonstrating an active icon in the Transform category.

Expression Blend provides an Event viewer mode so you can hook up controls that you are creating in Blend to code-behind file that let you go off and dynamically work out what to do next.

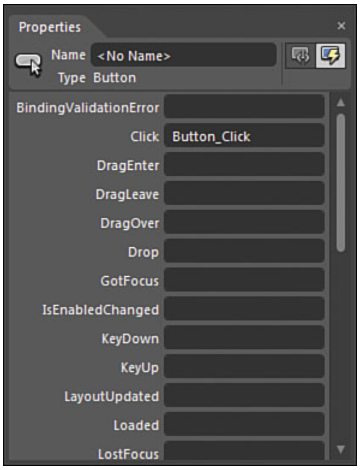
The key word here is “dynamic.”

The user interface layer you are creating in Blend is not dynamic at all. It might appear to be that way when you start animations and add effects, but the truth is that Expression Blend is simply creating a static mark-up language for you (called XAML), which defines what objects are where and how you might want to use those objects.

The code layer is very different to that as you can work with conditional logic, add and remove elements, and do all sorts of magical things.

Think of the user interface layer in Blend being a static design time layer that is more akin to drawing on a board, and think of the code layer as being all that power when your application is running—so-called runtime.

Figure 2.80 shows the Event viewer mode for a Button type and as you can see, a value has been given for the Click event. This Click event is linked up in the code behind file waiting for the user to click the button.



BindingValidationError	
Click	Button_Click
DragEnter	
DragLeave	
DragOver	
Drop	
GotFocus	
IsEnabledChanged	
KeyDown	
KeyUp	
LayoutUpdated	
Loaded	
LostFocus	

FIGURE 2.80 The Button event collection with specifically the Click event being subscribed to.

Don’t be overwhelmed by this concept, as it will be explained a little further in Chapter 3, “Using Expression Blend for the First Time” where you connect all the pieces together, including some simple events.

Resources Panel

The Resources panel contains a map to all the control element and property value mappings that are collectively referred to in Expression Blend as resources. Figure 2.81 provides an overview of the panel showing two resources in differing scenarios.

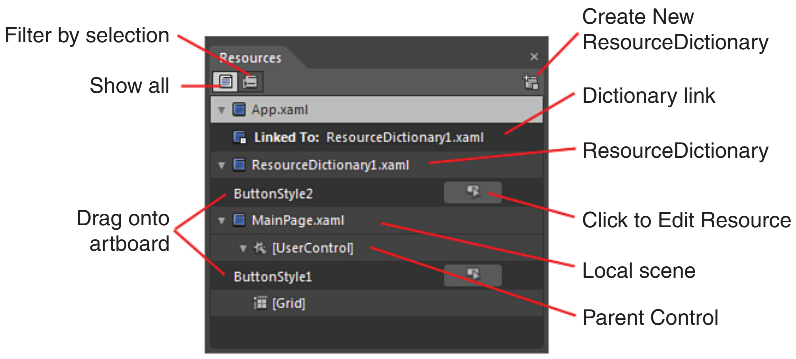


FIGURE 2.81 The Resources panel.

A concept that is key to using resources is that of the Resource Dictionary (RD), which is basically an XAML file that contains a collection of resource elements. You can push pretty much any type of resource into an RD, and when you edit any property value in the Properties panel, you can select the Advanced Options button; if applicable, you will be able to select “Convert to New Resource….”

When using an RD, the collections contained within it become accessible across your entire solution, so for example, you can create a default button style that you want to use everywhere and just by drag and dropping an element straight out of the Resources panel, your new button instance will appear. Any changes made to that style will be updated instantly across every button that references the style you have used.

As you can clearly see in Figure 2.81, there are two button styles added to this project. ButtonStyle1 is located as a child element of the [UserControl] element, whereas ButtonStyle2 is a child of ResourceDictionary1.xaml. This means that if you open the Assets panel and look at the Style collection, you see that only ButtonStyle2 is available as a global style, as shown in Figure 2.82.

Figure 2.83 shows that I am now filtering my resource collection by the current control that I am working on, so the Resources panel shows me the ButtonStyle1 style that I don’t have in the RD.

Chapter 13 will help you understand in greater detail how to work with the Resources panel and Resource Dictionaries.

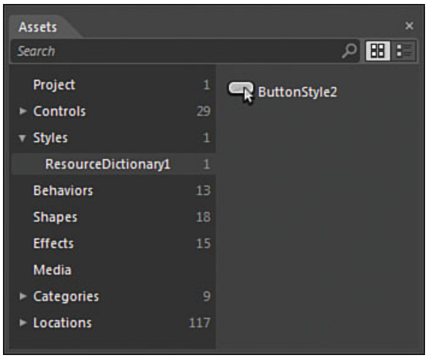


FIGURE 2.82 The ButtonStyle2 style available globally.

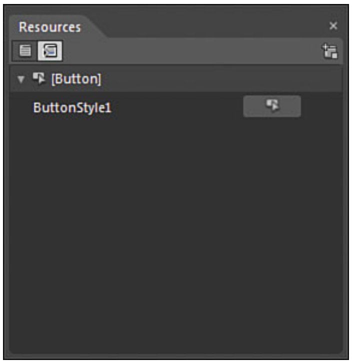


FIGURE 2.83 The ButtonStyle1 element available locally.

Data Panel

The Data panel contains a large amount of functionality specifically designed to get you working with structured and strongly typed data. This means that you can be sure of the data that you are working with, regardless of it being in designtime or runtime. Expression Blend enables you to create sample live data, which lets you style and present data in a meaningful way, all without having to write any code or get anyone else involved with providing data to your solution.

Figure 2.84 shows a sample data collection that has been created just for design time. You can see that the data collection is in use because of the orange binding border notification. The bottom half of the Data panel is also showing that the same data collection (in this instance, named SampleDataSource) is also providing the DataContext to the current scene. In essence, this means that any control I add to the artboard could get its display value and other settings directly from the DataContext.

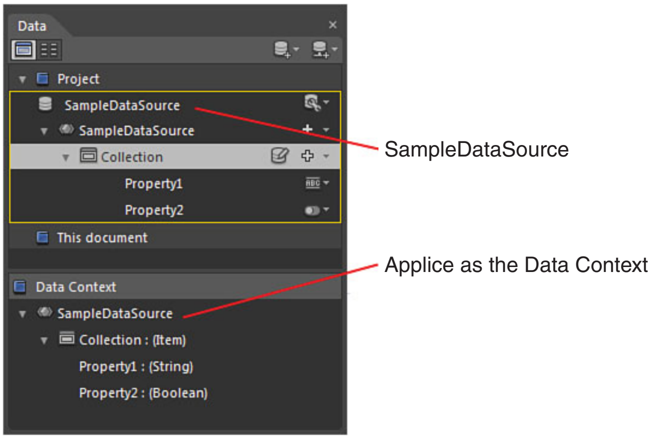


FIGURE 2.84 The Data panel in use.

Granted that if you haven't worked with data collections before, the preceding information probably sounds very confusing and very complex, but Chapter 10, "Expression Blend Data Support," shows you just how simple it is to work with.

Summary

This is one of the largest chapter of the book, and you have covered quite a lot of ground to this point—some very high-level views of certain areas of working with Expression Blend and some interesting discoveries to help you position the tool in your mind. You now have seen the vast majority of the panels and understand that you can set up Blend to work how you want and need it to.

Chapters further on from this one are about digging into the detail of all of these concepts, teaching you as easily as possible about how Expression Blend works.

This page intentionally left blank

Index

How can we make this index more useful? Email us at indexes@samspublishing.com