Microsoft Expression Blend and Silverlight for Windows Embedded

Writers: Frankie Anderson, Janet Fisher

Technical Reviewer: Natalie Cervantes

Published: March 2011

Applies To: Expression Blend 3 and Windows Embedded Compact 7

Abstract

Describes the workflow for Windows Embedded Compact 7 designers and developers working together to create Silverlight for Windows Embedded applications.

- Expression Blend workspace files in Silverlight for Windows Embedded
- Expression Blend and Silverlight for Windows Embedded project workflow
- Supported controls and features in Expression Blend
- Unsupported controls and features in Expression Blend

© 2011 Microsoft. All rights reserved.

Contents

Introduction 3

Expression Blend Workspace Files in Silverlight for Windows Embedded 3
- Silverlight for Windows Embedded Application Project Template 4
- Windows Embedded Silverlight Tools 5

Expression Blend and Silverlight for Windows Embedded Project Workflow 6
- Demonstrating Your Design 6
- Project Under Source Control 6

Supported Controls and Features in Expression Blend 7
- XAML 7
- States 7
- Storyboards 8
- Events 8
- Styles and Templates 9
- User Controls 9
- Win32Control 9

Unsupported Controls and Features in Expression Blend 9
- Managed Code 10
- Media 10
- Font Embedding 11
- Data Binding 11
- Effects 11
- Behaviors 12
- Custom Controls 12
- Event Handler Methods 12
- SketchFlow 12

Conclusion 13

Additional Resources 13

© 2011 Microsoft. All rights reserved.

Introduction

Microsoft Expression Blend 3 and Silverlight for Windows Embedded, combined with Microsoft Visual Studio, offer an end-to-end development environment. Windows Embedded Compact 7 device designers and developers can use this environment to create first-class, cutting-edge application interfaces in a wide range of device categories. Expression Blend gives designers a wide variety of visual tools to use to create dynamic Silverlight for Windows Embedded applications. Visual Studio gives developers a complete set of tools for modifying the native C++ code needed to drive embedded applications.

Designers and developers can efficiently work on a project in parallel using Expression Blend and Visual Studio respectively. Designers use the Expression Blend tool set to create a dynamic and compelling XAML UI design, while developers implement the corresponding business logic using C++ in Visual Studio. A designer works on the appearance and interaction of an application by drawing and modifying objects on the design surface in Expression Blend. A developer implements the underlying logic that interacts with the elements defined by the XAML files, using Visual Studio projects that contain C++ code files and resources derived for the XAML. The XAML elements are attached to events in the C++ code files to complete the application. Each change the designer makes automatically modifies the underlying XAML and managed code files in the Expression Blend project. The developer periodically imports the XAML changes and additions into Visual Studio project using Windows Embedded Silverlight Tools.

Expression Blend is intended to simplify user interface design by working like a paint program, but with features that let you easily incorporate animation and interaction designs with visual appearance. With Expression Blend, you can quickly create a look for your application that is unique to your company or organization. What you see on the design surface in Expression Blend is what your users will see when they run the final application.

This article gives an overview of how to use Expression Blend with the **Silverlight for Windows Embedded Application** project template and how to use Windows Embedded Silverlight Tools in Visual Studio to create Silverlight for Windows Embedded applications. For information about converting an Expression Blend project to a Silverlight for Windows Embedded Compact project in Visual Studio, see A Sample Application Tutorial Using Windows Embedded Silverlight Tools (http://go.microsoft.com/fwlink?linkid=189508) on MSDN.

Expression Blend Workspace Files in Silverlight for Windows Embedded

A designer creates the look and interaction of an application using the Expression Blend workspace. Changes the designer makes in the appearance of application elements or interactions with the elements are automatically propagated from the Expression Blend workspace to underlying XAML files

© 2011 Microsoft. All rights reserved.

and managed code (C# or Microsoft Visual Studio.NET) files. In Silverlight applications that run on Windows desktop computers and in Windows Presentation Foundation (WPF), these underlying XAML and managed code files can be used directly. However, because Silverlight for Windows Embedded supports a subset of XAML features, XAML files created in Expression Blend can only be used if they contain supported features. Similarly, Expression Blend managed code files cannot be used directly in Silverlight for Windows Embedded applications because those applications require native C++ code.

The **Silverlight for Windows Embedded Application** project template and the Windows Embedded Silverlight Tools are used to convert the standard underlying Expression Blend project files to Visual Studio native C++ project files for direct use in Windows Embedded Compact 7 applications. For information about converting an Expression Blend project to a Silverlight for Windows Embedded project in Visual Studio, see A Sample Application Tutorial Using Windows Embedded Silverlight Tools (http://go.microsoft.com/fwlink?linkid=189508) on MSDN.

Silverlight for Windows Embedded Application Project Template

By default, Expression Blend includes project templates for creating Windows Presentation Foundation (WPF) applications for the desktop or Silverlight applications for the web. To create Silverlight applications for Windows Embedded Compact in Expression Blend, you need to use the **Silverlight for Windows Embedded Application** project template. This template hides some controls in the Expression Blend toolbox and other features that are not compatible with the Silverlight for Windows Embedded API.

If you have Expression Blend 3 installed, you can get the **Silverlight for Windows Embedded Application** project template by selecting the **Windows Embedded Silverlight Tools** option when you install Windows Embedded Compact. The template is installed in one of the following folders:

- On a 64-bit computer: %ProgramFiles(x86)%\Microsoft Expression\Blend 3\Project Templates\en\Windows Embedded Silverlight Tools\Application
- On a 32-bit computer: %ProgramFiles%\Microsoft Expression\Blend 3\Project Templates\en\Windows Embedded Silverlight Tools\Application

If you do not want to install Windows Embedded Compact 7, you can get the **Silverlight for Windows Embedded Application** project template by going to the installation point (http://go.microsoft.com/fwlink/?linkid=199902) on MSDN.

You must manually copy the template files from %ProgramFiles%\Microsoft Expression\Expression Blend 3\Project Templates\en\Windows Embedded Silverlight Tools\Application to your local user directory at

%Users%\<user-name>\Documents\Expression\Blend 3\Project Templates\Windows Embedded Silverlight Tools

so that Expression Blend can display the templates in its **New Project** dialog box. Note that if the local user directory does not already exist, you must create it prior to copying the template files into it.

© 2011 Microsoft. All rights reserved.

To use the project template to create a new Windows Embedded Compact project in Expression Blend, click the **Windows Embedded** project type under **Silverlight** in the **New Project** dialog box, then click **Silverlight for Windows Embedded Application**, and then choose your project name and location.

Note

If you create a project using a template other than the **Silverlight for Windows Embedded Application** template, your project will include XAML elements or features that are not supported in a Silverlight for Windows Embedded project. Your XAML files will not display in **Design** view in Visual Studio, and your application will not run on your Windows Embedded Compact device until all of those unsupported features are removed. For a list of unsupported features, see the section Unsupported Controls and Features in Expression Blend in this article.

Windows Embedded Silverlight Tools

In an Expression Blend project, XAML files are created with corresponding code-behind files. A code-behind file defines a partial class that is completed by the XAML file. Code-behind files in Expression Blend are created using managed code (C# or Visual Basic.NET) that Silverlight for Windows Embedded does not support. There are two scenarios you can use to convert an Expression Blend project for use in Silverlight for Windows Embedded. Both of these scenarios use Windows Embedded Silverlight Tools, a suite of developer tools that integrates with Visual Studio 2008 to automate and expedite the process of creating native (C++) Windows Embedded Compact 7 applications from Silverlight UI designs created in Microsoft Expression Blend 3. You also use Windows Embedded Silverlight Tools to update your Visual Studio project from your Expression Blend project as additions and changes are made.

The first scenario for converting an Expression Blend project is to use Visual Studio to create a new Platform Builder project. You then import the Expression Blend project into a subproject by using Windows Embedded Silverlight Tools. Refer to A Sample Application Tutorial Using Windows Embedded Silverlight Tools (http://go.microsoft.com/fwlink?linkid=189508) on MSDN for details on using the wizard.

The second scenario for converting an Expression Blend project is to use Visual Studio to create a Smart Device project and then import the Expression Blend project into it. Note that this scenario requires that you install a Silverlight for Windows Embedded SDK on your computer. Refer to A Sample Application Tutorial Using Windows Embedded Silverlight Tools (http://go.microsoft.com/fwlink?linkid=189508) on MSDN for details on using the wizard and creating and using the SDK.

© 2011 Microsoft. All rights reserved.

Expression Blend and Silverlight for Windows Embedded Project Workflow

For designers and developers to efficiently work together, they must agree in advance upon a designer/developer contract. This contract defines the interface between the visual aspects of the UI that are implemented in the XAML files and the business logic, in addition to the event handling aspects that are implemented in the code-behind files.

After an initial contract is worked out, the designer creates prototype UI designs in Expression Blend. The designer uses the guidelines in the contract to choose and name controls, determine where to use placeholder user controls and placeholder Win32Control controls, when to use storyboards versus visual states, and when to use event handlers.

After the design team chooses an initial prototype design, developers use Visual Studio and Windows Embedded Silverlight Tools to convert and import the design to a Silverlight for Windows Embedded project so that the development team can concurrently move forward on business logic and event handling tasks. The designer continues to modify the design in Expression Blend, and the developer uses the Visual Studio project to implement and modify C++ classes for the application. To incorporate UI design changes into the application, the developer periodically uses Windows Embedded Silverlight Tools to update the Visual Studio project from the Expression Blend project.

For information about the designer/developer contract and workflows, see Introduction to Application Development with Silverlight for Windows Embedded (http://go.microsoft.com/fwlink?linkid=183252).

Demonstrating Your Design

Expression Blend is a valuable prototype demonstration tool as well as a visual design tool. During the design process, you can press F5 to run your **Silverlight for Windows Embedded Application** project in a browser to see how it will appear on your Windows Embedded Compact device. This is useful if you want to present numerous designs in different projects before handing off the final design to a developer.

Project Under Source Control

If your team uses a source control system, you should add both the Expression Blend project and the converted Visual Studio project to source control.

Note

We highly recommend using a source control system. A source control system can include features such as notification when team members change files, file merging upon check-in, and reverting change lists if issues arise.

If an Expression Blend project is under source control using Microsoft Visual Studio Team System 2008 Team Explorer, you can use the context menus of Expression Blend to check project files in and out.

© 2011 Microsoft. All rights reserved.

For information about using the source control features of Expression Blend, see Working with Team Foundation source control (http://go.microsoft.com/fwlink?linkid=183253) on MSDN.

Supported Controls and Features in Expression Blend

In general, if a control in the Expression Blend toolbox or another feature is not compatible with Silverlight for Windows Embedded, the **Silverlight for Windows Embedded** project template hides it. The following sections describe the controls and features that are supported in a Silverlight for Windows Embedded project in Expression Blend. Note that the Silverlight for Windows Embedded API supports more features than the **Silverlight for Windows Embedded** project template exposes in Expression Blend, but those features must be manually applied in Visual Studio.

XAML

When you use the visual design tools of Expression Blend, the XAML of your application is automatically modified. For example, when you select the **Button** tool from the **Toolbox** or from the **Assets** panel, and you use your cursor to draw a button on the artboard in the Expression Blend workspace, XAML similar to the following is added to the file in which you are working:

```
<Button
    Height="26"
    HorizontalAlignment="Right"
    Margin="0,0,63,206"
    VerticalAlignment="Bottom"
    Width="45"
    Content="Button"
/>
```

When you use Windows Embedded Silverlight Tools to convert your Expression Blend project to a Windows Embedded Compact project in Visual Studio, your XAML is preserved, and the XAML files in your Expression Blend project are converted to resources in the **Resources** folder of the final Visual Studio project.

States

In Expression Blend, you can define a different visual appearance for each state that a control to which you can apply templates, or a user control, can be in. For example, the template of a **Button** control includes a **Pressed** state that darkens the color of the button when a user clicks it. You can modify the

© 2011 Microsoft. All rights reserved.

transition time between two states, and even create animation storyboards that run when a state is first entered.

When you convert your project in Visual Studio, any state definitions and transitions in the XAML files of your Expression Blend project are preserved.

For information about using states and transitions in Expression Blend, see Defining different visual states for a control (http://go.microsoft.com/fwlink?linkid=183255) on MSDN.

Storyboards

In Expression Blend, you can create animations by modifying objects and setting key frames in a storyboard. Each key frame defines a visual state for one part of a storyboard animation. Storyboards can be stand-alone animations that are triggered to run when a user interacts with your application, or they can be part of a state group that animates the transition between two states.

When you convert your project in Visual Studio, storyboards that are defined in XAML are preserved. However, in your converted Silverlight for Windows Embedded project, you must use the **IXRFrameworkElement::FindName** method to obtain a smart interface pointer for the storyboard object to run it from a method in your code. For example, you might use the following code to obtain an **IXRDoubleAnimationPtr** to a storyboard named SecondAnimation:

```
IXRDoubleAnimationPtr pSecondAnimation = NULL;

pRoot->FindName(L"SecondAnimation", &pSecondAnimation);

pSecondAnimation->Begin();
```

For information about using storyboards in Expression Blend, see Animating objects (http://go.microsoft.com/fwlink?linkid=191158) on MSDN.

Events

You can attach event handler methods to objects in your Expression Blend application by using the **Events** view of the **Properties** panel. Stubs for the event handler methods are automatically added to your code-behind files.

You can write code for those event handler methods to make your prototype application in Expression Blend more interactive, but when you convert your application in Visual Studio, that .NET Framework code is ignored, and template Visual C++ code stubs are generated for your event handler methods. For more information, see Managed Code in the section Unsupported Controls and Features in Expression Blend.

For information about using events in Expression Blend, see Create a new event handler method (http://go.microsoft.com/fwlink?linkid=183256) on MSDN.

© 2011 Microsoft. All rights reserved.

Styles and Templates

You can apply modified styles and templates to your controls in Expression Blend to give your application a unique or themed appearance.

When you convert your project in Visual Studio, any styles and templates defined in the XAML files of your Expression Blend project are preserved.

For information about using styles and templates in Expression Blend, see Styling a control that supports templates (http://go.microsoft.com/fwlink?linkid=183258) and Styling tips for common Silverlight controls (http://go.microsoft.com/fwlink?linkid=183257) on MSDN.

User Controls

You can create your own custom control by creating a new **UserControl** item in Expression Blend. A user control is stored in a XAML and code-behind file, just like the main page of your application. Templates cannot be applied to user controls.

When you convert your project in Visual Studio, wrapper classes are generated for each user control defined in XAML, and the user controls are automatically registered in the **App::RegisterUserControls()** method.

Visual C# code in the code-behind file for a user control is ignored, but empty native Visual C++ stubs are created for methods. For more information, see Managed Code in the section “Unsupported Controls and Features in Expression Blend.”

For information about creating user controls in Expression Blend, see Designing a user control from scratch (http://go.microsoft.com/fwlink?linkid=183259) on MSDN.

Win32Control

The **Win32Control** placeholder control is provided with the Windows Embedded Silverlight Tools. It provides a way to combine standard Win32 controls with new Silverlight for Windows Embedded controls. This control is available from the **Assets** panel in Expression Blend. You can use it to represent the future location of a graphics device interface (GDI) control in the final Visual Studio project. GDI controls are child windows that will always appear in front of other objects in your Silverlight for Windows Embedded application.

Unsupported Controls and Features in Expression Blend

This section describes the controls and features that are not supported in a Silverlight for Windows Embedded project in Expression Blend, but that are supported in Visual Studio.

© 2011 Microsoft. All rights reserved.

Managed Code

When you press F5 in Expression Blend to run your project in a browser window, you might want it to look and behave as much like the final Windows Embedded Compact application as possible. For example, to demonstrate the interactivity of your design in Expression Blend, you might write Visual C++ code for event handler methods. You might also include Visual C# classes in your project for elements like custom UI controls or effects.

However, any Visual C# code, either in code-behind files (which contain managed code for elements defined in XAML) or in custom class files, is not converted when you use the Windows Embedded Silverlight Tools to convert your Expression Blend project to a Windows Embedded Compact project in Visual Studio. Instead, empty native C++ stub code is generated for each of your Visual C# methods and classes.

Important

If you want empty stub code to be generated for your Visual C# event handlers, you must explicitly name the XAML element that uses the event handler method in your Expression Blend project. For example, the following XAML code shows a button that is named **MyButton**.

```
<Button
    x:Name="MyButton"
.../>
```

If you are using the Windows Embedded Silverlight Tools to update a project that has already been converted once, empty stub code is generated only for new Visual C# methods and classes that don't already exist by the same name in the Visual Studio project.

After a project goes through an initial conversion, designers and developers can work at the same time. For example, a developer can write a native Visual C++ event handler method named **_Button_Click** in the Visual Studio project, and a designer can create a new button object and connect it to an event handler method named **_Button_Click** in the Expression Blend project. When the project is updated in Visual Studio, the new button element is already connected to the native Visual C++ **_Button_Click** method. Using the same names for objects and event handler methods is part of the designer/developer contract. For more information, see the section Expression Blend and Silverlight for Windows Embedded Project Workflow in this article.

Media

Microsoft Silverlight desktop and web applications in Expression Blend can use a **MediaElement** object to play video and audio files. However, **MediaElement** is not supported in a Silverlight for Windows Embedded project. You can still use Expression Blend to design the UI for a media player, for example, by designing a unique look for the playback buttons. After your project is imported into Visual Studio, a developer can create and position a DirectX content window that displays the media. If your UI is positioned on top of the DirectX content window in your final project, it appears in front of the window.

© 2011 Microsoft. All rights reserved.

Font Embedding

Silverlight desktop and web applications in Expression Blend can use font embedding to include specific fonts or subsets of fonts in the application that will run on a computer that doesn't already have the font installed. Font embedding ensures that your application will display exactly the way you designed it.

However, font embedding is not supported in a Silverlight for Windows Embedded project in the same way. To embed a font in a Silverlight for Windows Embedded application, add the font (.ttf or .ttc) file to the project resource (.rc) file as a "XAML_RESOURCE" resource. Then set the **FontFamily** property to "<font file name>#<font name>". For example, in your XAML file, you could use the following element:

```
<TextBlock>
    <Run FontFamily="MyFont.TTF#MyFontName" Text="This text uses a different font face."/>
</TextBlock>
```

The **FontFamily** property also supports absolute or relative paths such as "/My Documents/MyFont.TTF#MyFontName", but the path cannot contain a colon (:), two backslashes (\\), or two forward slashes (//).

Note

If you decide to embed, create a subset of, or otherwise redistribute fonts in your application, it is your responsibility to make sure that you have the required license rights for those fonts.

Data Binding

Data binding is not supported in Expression Blend when you are working on a Silverlight for Windows Embedded project. The **Data** panel is not available, nor is **Data binding** available from the **Advanced property options** context menu.

Note

The **DataContext** property is not hidden in the **Properties** panel of Expression Blend.

However, data binding features are supported in a Silverlight for Windows Embedded project if they are added manually in Visual Studio, and some data binding features are supported in XAML if they are added manually in **XAML** view of Expression Blend. For information about which data binding features Silverlight for Windows Embedded supports, see “Differences between Silverlight for Desktop Browser and Silverlight for Windows Embedded” in Windows Embedded Compact Help.

Effects

Expression Blend comes with two graphic effects: the blur effect and the drop shadow effect.

© 2011 Microsoft. All rights reserved.

These graphic effects are provided when you are working on a Silverlight for Windows Embedded project to help you make your prototype design more appealing. However, when you convert your Expression Blend project in Visual Studio, you cannot load and parse graphic effects created by using XAML. You must create custom effects using the **IXRCustomEffect** Visual C++ class, and apply them using the **IXREffect** Visual C++ class.

For information about how effects are implemented differently in Silverlight for Windows Embedded, see “Differences between Silverlight for Desktop Browser and Silverlight for Windows Embedded” in Windows Embedded Compact Help.

Behaviors

Behaviors are reusable pieces of packaged code that can be dragged onto any object, and then fine-tuned by changing their properties. They add interactivity to your application without the necessity of writing code.

Behaviors are not supported in a Silverlight for Windows Embedded project.

Custom Controls

Custom controls are not supported in a Windows Embedded Compact project in Expression Blend. Use the **Win32Control** placeholder control in Expression Blend to represent the location of a custom control in the final Visual Studio project. Custom triggers, actions, and behaviors also are not supported.

However, a developer might provide you with a .NET version of a custom object that you can use when prototyping your application in Expression Blend. When the project is imported into Visual Studio, the code for custom objects is ignored.

Event Handler Methods

You can connect event handler methods to objects in your Expression Blend application by using the **Events** view of the **Properties** panel. Stubs for the event handler methods are automatically added to your code-behind files.

You can write code for those event handler methods to make your prototype application in Expression Blend more interactive. However, when you convert your application in Visual Studio, that .NET Framework code is ignored, and empty native Visual C++ stub code is generated for your event handler methods. For more information, see the Managed Code section above.

SketchFlow

SketchFlow is a feature of Expression Blend that provides extra functionality for the prototyping process. You can use it to draw a map of the structure of your application, animate possible user interaction sequences, and gather feedback using notations in your designs.

SketchFlow projects cannot be converted by the Windows Embedded Silverlight Tools in Visual Studio. However, you can manually copy and paste XAML elements from a SketchFlow project into a new

© 2011 Microsoft. All rights reserved.

Silverlight for Windows Embedded project in Expression Blend. Some elements must be converted to compatible versions of those elements. For example, controls such as **TreeView**, **TabControl**, and **Menu** must be prepended with the **xrex** namespace identifier, as in the following example:

```
<xrex:TreeView .../>

<xrex:TabControl .../>

<xrex:Menu .../>
```

The namespace is defined at the top of a XAML file in a Silverlight for Windows Embedded project in Expression Blend, as shown in the following example:

```
<UserControl

    xmlns:xrex="clr-
namespace:EmbeddedXamlRuntimeExtension;assembly=EmbeddedXamlRuntimeExtension" ...>
```

Conclusion

Windows Embedded Compact device designers can use Expression Blend and Silverlight for Windows Embedded to design and prototype a rich user interface in embedded devices. Using Expression Blend you can easily build and modify applications and see exactly how they will appear to the end user before the applications are implemented in detail. The Expression Blend environment separates the visual elements from the programming logic so that the visual designer and the logic developer can work on a project in parallel, after initially determining the interface between the UI and the application implementation, and the responsibilities of the two roles.

Expression Blend supports some features that are not available to Silverlight for Windows Embedded applications, and the Silverlight for Windows Embedded Application project template is used to hide incompatible elements so that they are not available for use. The C# or Visual Basic code-behind files that are automatically generated inside an Expression Blend project are not directly usable by Windows Embedded Compact applications, so Visual Studio uses Windows Embedded Silverlight Tools to convert the Expression Blend code files to files that contain appropriate native C++ code stubs. A developer then uses Visual Studio to modify the code files to provide the business logic for the application. The Visual Studio project combines the native C++ code files with the XAML from Expression Blend to complete the Silverlight for Windows Embedded application.

Additional Resources

For more information about Expression Blend, see the following resources:

© 2011 Microsoft. All rights reserved.

Expression Blend User Guide, available when you press F1 in Expression Blend, or available on MSDN at Expression Blend 3 (http://go.microsoft.com/fwlink?linkid=204757). Note that the User Guide does not include information specific to Windows Embedded Compact. For information about the differences between developing Silverlight applications and developing Silverlight for Windows Embedded applications, see “Differences between Silverlight for Desktop Browser and Silverlight for Windows Embedded” in the Windows Embedded Compact Help.

- Microsoft Expression Community website (http://go.microsoft.com/fwlink?linkid=104337). This site includes links to blogs, forums, and video tutorials like those at Expression Blend Training Videos (http://go.microsoft.com/fwlink?linkid=181255).
- Windows Embedded website (http://go.microsoft.com/fwlink/?LinkId=183524)

© 2011 Microsoft. All rights reserved.

This document is provided “as-is.” Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

© 2011 Microsoft. All rights reserved.

© 2011 Microsoft. All rights reserved.