Windows® Embedded

Introduction to Application Development with Silverlight for Windows Embedded

Windows Embedded CE 6.0 R3 Technical Article

Writers: David Franklin

Published: September 2009

Applies To: Windows Embedded CE 6.0 R3

Abstract

Windows Embedded Compact now includes Silverlight for Windows Embedded, with which designers and developers can create exciting and visually compelling user interfaces for embedded devices. Silverlight for Windows Embedded makes it much easier to produce a compelling user experience than was previously possible. The design and development approach used to create these user interfaces is very different from the Win32-based approach that most designers and developers are familiar with. This whitepaper describes how to go from a prototype user interface design to a working Silverlight for Windows Embedded project.

The key benefit of the approach described in this article is that the designers and developers are able, for the most part, to work independently of each other. Designers and developers can then focus on the pieces of the overall project that are in their area of expertise, using the tools that are most familiar to them.

For example, once the Expression Blend project is set up, the designer works in Expression Blend producing XAML (eXtensible Application Markup Language) and doesn't spend time figuring out what the developer needs to do to hook the XAML to the programming logic in the separate code-behind file. Microsoft Expression Blend is a tool for creating graphics, animations, and user interfaces, which uses XAML and the same project system as Visual Studio. This allows designers and developers to share the same files and collaborate.

To enable this independent work style, the designer and developer work together to create the Designer/Developer Contract, which defines the interface between the UI layer that the designer works on and the business logic layer that the developer works on. This Contract serves as the basis for the initial Expression Blend project the designer uses and the initial C++ implementation that the developer creates.

Introduction

Silverlight for Windows Embedded gives designers and developers the functionality they need to create exciting and visually compelling user interfaces for embedded devices. The design and development approaches used to develop these user interfaces are very different from the Win32-based approaches that most designers and developers are familiar with.

This article describes how to go from a prototype user interface design to a working Silverlight for Windows Embedded project. The key benefit of this approach is that once the project implementation begins, the designers and developers can work in parallel with minimal interaction.

There are two primary roles in the Silverlight for Windows Embedded implementation process:

- The designer, whose tools include graphics editors for making visual assets and Microsoft Expression Blend for creating the XAML that defines the layouts and animations in the device user interface
- The developer, whose tools include Visual Studio and Platform Builder

For designers and developers to work efficiently together it is critical that the designer is able to work on the visual aspects of the project at the same time the developer works on the underlying business programming logic. To facilitate this, the designer and developer work together to create a Designer/Developer Contract that defines the interface between the UI layer and the business logic layer.

This Contract and prototype are used to create a skeleton project in Expression Blend for the designer and a set of headers and implementation files for the developer. To guide the designer in transforming a prototype into an Expression Blend project, there are several design patterns available. The developer then uses the Expression Blend project to create a corresponding set of C++ header and implementation files.

After the prototype is created, the designer and developer can work independently of each other on their individual portions of the project. If they both follow the Contract, they can assemble the XAML from the Expression Blend project and the C++ files at any time and view the current state of the user interface.

The Contract will probably have to be refined during the course of a project as the designer improves upon the prototype. In such situations, the designer and developer simply make the necessary changes to the Contract, reflect those changes in the Expression Blend project and C++ files, and then continue on with their independent implementation efforts.

Design and Development Working Together

The key to the approach for the user interface implementation described in this paper is that the two parties in the process, the designer and the developer, have a clear division of responsibilities. Designers work on their portion of the project using the tools they prefer and the developers work on their portion of the project using the tools they prefer. They both can work without worrying about whether their work is going to break what the other party is doing.

Throughout this document, the two parties are referred to as “the designer” and “the developer.” In nearly all projects, there will be more than one designer or more than one developer, but for the sake of simplicity the collection of designers are referred to as “the designer” and the collection of developers are referred to as “the developer.”

The designer’s first responsibility is to create the initial user interface prototype. Then the designer works with the developer to write the Contract, based on this prototype. Once the Contract is complete, the designer creates an Expression Blend project that is a skeleton of the prototype, at the minimum covering everything detailed in the Contract. Finally, the designer works in Expression Blend to refine the design.

The developer’s first responsibility is to work with the designers to write the Contract. Depending on the background of the designer, the developer may need to help create the initial Expression Blend project. Then the developer uses the Contract to create the C++ files that connect with the XAML files.

Table 1 provides a list of responsibilities for each of the key roles in application development with Silverlight for Windows Embedded.

Table 1: Key Roles in Application Development with Silverlight for Windows Embedded

Role	Responsibilities
The Designer	Initial UI prototyping Coauthors the Contract Creates visual elements Uses Expression Blend to author XAML
The Developer	Coauthors the Contract Uses Visual Studio to write the underlying business programming logic Uses Visual Studio to write the code that connects the business programming logic to the XAML Takes the XAML from the designer and builds the end-to-end project

Starting a Project

One common problem that user interface designers make is starting to write the “real” implementation too early. It is critical, especially when designing the user interface for a new product, to spend adequate time prototyping for the following reasons:

- Prototypes allow designers to quickly try out numerous possible designs and focus in on their favorites.
- Some level of usability testing can be performed on prototypes and the cost of fixing usability issues during prototyping is much less than during the project implementation phase. Usability testing often leads to new design ideas that might not otherwise have been found.
- Prototypes give a development team a tangible example to strive for.
- The less mature a project concept is before writing the initial Designer/Developer Contract, the more likely that it will need to be extensively refined during the course of application development.

Prototyping Tools

There are numerous prototyping tools available to a designer and numerous excellent blogs, papers, and books describing how to use them. Designers should use whatever prototyping tools they are most comfortable with. Common tools include pencil and paper, Microsoft PowerPoint, or a graphics editor. The latest version of Microsoft Expression Blend includes Sketchflow, a powerful prototyping tool that allows designers to quickly create screen layouts and connect the screens together into a functional prototype.

The Designer/Developer Contract

Once the designer is happy with the structure of the prototype for the application, the designer and the developer need to work together to define how the UI layer and the underlying business programming logic (also called the code-behind) fit together. This is referred to as the Designer/Developer Contract, or, simply, the Contract. This Contract needs to cover the following items:

- **User Interface Controls**: Any text boxes, edit boxes, check boxes, radio buttons, list boxes, or other controls that need to be accessed in the code-behind need to have agreed-upon names. This allows the code-behind to access the state of these items and take appropriate actions. A useful convention is to prefix the names of these controls with an underscore. For example, a list box that contains a user's contracts could be named "_contacts".
- **UserControls**: In Expression Blend, designers can encapsulate multiple user interface controls into entities called UserControls. These can cover concepts as simple as a set of strings making up a contact card, or as complex as a screen in the user interface. The code-behind defines the functionality for each of these UserControls as a C++ class and so the Contract must specify the class name of this UserControl.
- **Storyboards or Visual States**: Expression Blend provides rich support for defining animations of various items in the user interface. Storyboards and Visual States (in the Visual State Manager) can be used to provide transitions between states in the design, as well as just moving things around the screen. In the code-behind, Storyboards and Visual States are referred to by name and so the names of these animations must be specified in the Contract.
- **Event Handlers**: User actions in the user interface are reported back to the code-behind as events. For example, when the user clicks on a button, a callback function associated with that button is called in the code-behind, allowing the application to take the appropriate action. Depending on how the developer attaches these callback functions to the user interface controls, the actual name of these callback functions may or may not be important. However, it is critical that the Contract enumerates all of the events that need to be handled in the user interface.

The Contract can be implicit in the structure of the Expression Blend project, but it is often helpful to write a document that explicitly states the control names, UserControls, Storyboard names, Visual States, and event handlers that are covered in the Contract.

It may be somewhat intimidating at the start to go from a rough prototype to the structure and rigor of the Designer/Developer Contract. The following guidelines are intended to help you create a Contract:

- Screens in the user interface should correspond to UserControls. This way the designer can easily see the layout of each screen in Expression Blend. If a number of screens share the same layout, it may make sense to have all of these screens correspond to a single UserControl. For example, if the project has multiple menu screens, they are likely to share a single layout.
- Groups of user interface items that function as a single entity, especially if they appear in multiple places throughout the UI, correspond to UserControls. For example, in a music playback application albums are likely to be displayed in lists in a particular way (for example, the album cover art on the left and the artist name and album name stacked on the right). Because this configuration will appear numerous times, it should be encapsulated in a UserControl.
- Simple user interface items like text, buttons, and checkboxes that do not have complicated interactions with other user interface items should correspond to the appropriate built-in controls, like TextBlock, Button, and CheckBox.
- Graphic assets, even if they are only placeholders, should be added to the initial Expression Blend projects. It is easy to change them later.
- If a UserControl has multiple modes that are visually distinct, these modes can be represented as states in the Visual State Manager. In addition to obvious examples like button click states, the screens making up a wizard can be represented as Visual States, allowing visually rich transitions between the screens.
- For user interface visual transitions that don't naturally map to Visual States, Storyboards can be used to capture the transitions.
- Add event handlers for each type of event for any user interface item that responds to user input like mouse clicks, mouse drags, or text entry.

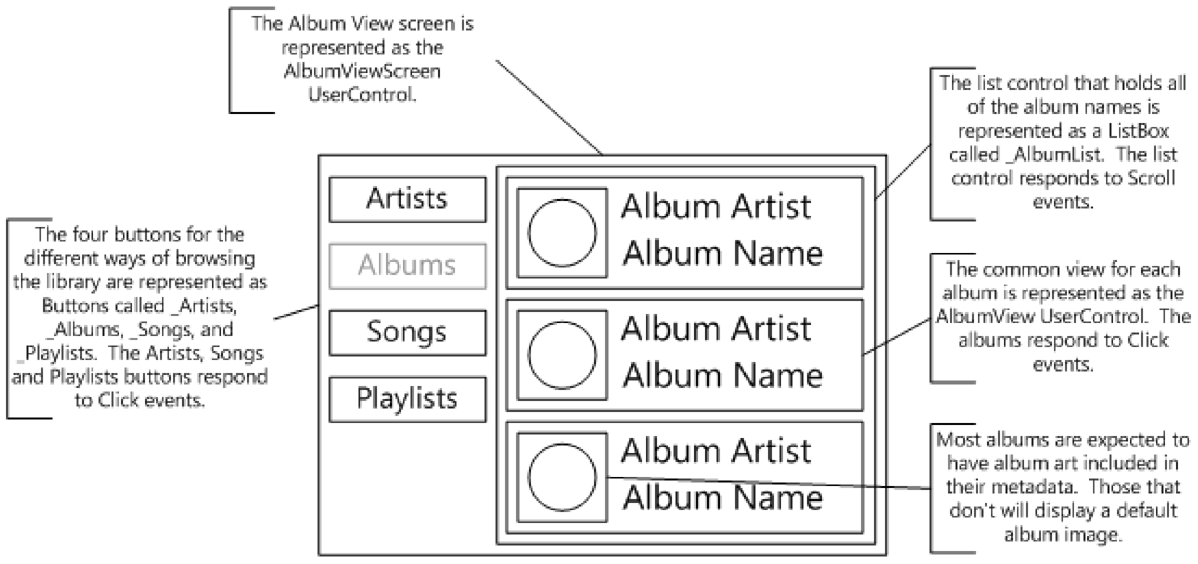
The Album View screen is represented as the AlbumViewScreen UserControl.
The list control that holds all of the album names is represented as a ListBox called _AlbumList. The list control responds to Scroll events.
The four buttons for the different ways of browsing the library are represented as Buttons called _Artists, _Albums, _Songs, and _Playlists. The Artists, Songs and Playlists buttons respond to Click events.
Artists
Albums
Songs
Playlists
Album Artist
Album Name
Album Artist
Album Name
Album Artist
Album Name
The common view for each album is represented as the AlbumView UserControl. The albums respond to Click events.
Most albums are expected to have album art included in their metadata. Those that don't will display a default album image.

Figure 1: An Example of the Contract Details for a Simple Prototype Screen

Making the Initial Expression Blend Project

With the Contract complete, the designer and developer can begin implementing the project.

To get started on the design side, the designer needs an initial Expression Blend project that contains all of the relevant details of the project. If the designer has a reasonable familiarity with software development, it may be feasible for the designer to create the Expression Blend project. Otherwise, we recommend that the designer and developer to work together to create the Expression Blend project.

To create the initial Expression Blend project:

1. In Expression Blend, create a new Project of type Silverlight Application.
2. For each UserControl that is a part of the project, select **Add New Item…** and add a UserControl with the desired name.
   - In the example in Figure 1, there are two UserControls to be added: **AlbumViewScreen** and **AlbumView**.
   - For projects with more than a handful of UserControls, it is advisable to add folders to the project and put the UserControls in the appropriate folders, so that they can be easily found. The file hierarchy needs to be agreed upon by both the designer and the developer.
3. For each UserControl, add the user interface items that are specified in the Contract.
   - In the example in Figure 1, the **AlbumViewScreen** UserControl needs five items: the Buttons (_Artists, _Albums, _Songs and _Playlists) and the ListBox (_AlbumList).
4. For UserControls that have Visual States associated with them, add each of the named states for the UserControl. Similarly, add Storyboards for all other visual transitions needed for the UserControl. The actual animations are added later.
5. For each user interface item that responds to events, add event handlers in the **Events** pane.
   - In the example in Figure 1, the Buttons respond to Click events.

One helpful way that a developer can ensure that the contract is maintained is to write a C# code-behind for the application that provides some functionality and touches each item called out in the contract. This code-behind can also be used to populate UserControls with values that allow the designer to see how they will appear.

Making the Initial C++ Project

Once the initial Expression Blend project is complete, the developer uses the Contract to create the C++ files that will connect with the XAML files from the designer. Each UserControl in the Expression Blend project requires a corresponding pair of C++ header and implementation files.

- The header file includes the class definition for the UserControl, member variables for each user interface item called out in the Contract, and any event handlers called out in the Contract.

- The source code file includes code stubs for instantiating the UserControl and for each event handler called out in the Contract.

Now you are ready to begin the implementation phase of your project.

Implementing and Iterating

The primary goal of the Designer/Developer Contract is to allow the designer and the developer to work in parallel. The Contract also allows the designer to send updated XAML files and assets to the developer and have them immediately work with the developer's code. If the developer has provided C# code-behind for the designer, the designer should always be quickly notified when the contract is broken and should be able to backtrack and fix the issue.

The designer's primary work during the implementation phase will include the following tasks:

- Refining the layout of screens and other UserControls. The initial version of the Expression Blend project consists only of the elements called out in the Contract. During implementation the designer will likely both refine the locations of these elements and add new visual elements like backgrounds and borders, which improve the aesthetics of the design, but not the actual function.
- Changing the visual appearance of the various user interface items. This includes choosing fonts, font sizes, border widths, and background gradients.
- Refining and creating graphic elements. Having excellent icons and other graphic elements can make the difference between a user interface that looks professional and one that looks amateur. The designer is likely to spend significant time creating and refining these assets.
- Creating the animations. Great Storyboard and Visual State transition animations can delight end-users and draw their attention to the user interface items that they need to see or act upon. Expression Blend allows very fine, timeline-based scripting of animations.

The developer's primary work during the implementation phase will include the following tasks:

- Writing the business logic that maintains the application's internal state and its interaction with external systems. This is the underlying functionality that the developer would need to write, regardless of what type of user interface the project has.
- Writing the logic that populates the XAML-based UI with the relevant application state. The business logic that the developer works on will have data and data structures that correspond with user interface elements in the XAML. In the example in Figure 1, the _AlbumList element is a ListBox that will contain one AlbumView UserControl for each album that is contained in the end-user's music collection. The business logic will provide some way of getting this collection. Then the developer will need to populate the _AlbumList with all of these albums.

Periodically, the designer and developer should take the XAML from the designer and the code-behind from the developer and see how the design looks and functions on the actual hardware. Early in a project you can do this infrequently, because the application won't be able to do much until the developer implements a significant amount of business logic. Later in a project you should test the design at least once a week. The benefits of doing so are as follows:

- Problems with the Contract are discovered early and can be resolved with minimal cost.
- The designer can see what the user interface looks like on the actual screen and how it performs on the actual hardware. This is very important on embedded devices, because the performance and display on a desktop PC is very different than the performance and display on an embedded device.
- The team can see the progress of the project. Regularly seeing the latest implementation running on the actual hardware can encourage a team and give management and other stakeholders confidence in the progress of the project.

Updating the Designer/Developer Contract

The designer usually has a lot of flexibility in refining the user interface while working within the Contract. However, it is very common that the designer will discover a compelling design idea that doesn't fit into the current Contract. In this situation, the designer and developer need to work together to make whatever changes are needed so that the designer's vision can be realized.

For simple changes the designer might have a particular change to the Contract in mind. For more complicated changes the designer may simply have an idea of the behavior that the user interface should have. In such a case the designer should work with the developer to figure out what changes to the Contract are needed to achieve the desired behavior.

After the Contract is revised and the developer and designer make the necessary changes to the Expression Blend project and C++ files, they can resume implementation.

Conclusion

Silverlight for Windows Embedded opens up many new options for user interface design for Windows Embedded Compact devices. User interfaces that were previously very difficult to implement can now be quickly designed and implemented. However, to take advantage of the new capabilities, designers and developers need to change how they implement the user interface. This article describes how a project team can go from a prototype user interface design to a working Silverlight for Windows Embedded project, using the Design/Developer Contract as a guide throughout the project.

The Contract specifies how the work that the designer is doing (authoring XAML with Expression Blend) fits together with the work that the developer is doing (writing C++ code with Visual Studio). A well-defined Contract stipulates the dependencies between the two parties so that each can work autonomously and each piece of the project will still fit together. This Contract is used to create the initial Expression Blend project and the initial C++ files that the designer and developer will each expand upon. If the designer and developer both carefully follow the Contract, the XAML from the designer and the C++ from the developer can be combined at any time to show the current state of the project.

This process allows the designers and developers to completely focus on the work that they do best while using the tools that they are most comfortable with.

For more information:

Windows Embedded Web site

Copyright