

Unni's Blog

Design, Expression Blend, Silverlight, WPF

<http://blogs.msdn.com/unnir/default.aspx>

Introducing sample data for Developers

Blend 3 brings to the table really sophisticated support for sample data, enabling better designability of your applications. In my previous [post](#), I demonstrated how designers could leverage this feature to prototype data connected applications inside Blend . However, for really sophisticated applications, richer support might be desirable - for example, you might want to use your own custom types for sample data in the cases where you cannot re-create the schema from scratch using the primitives that we provide.

Here is a small example that will help you get started on a feature that we just introduced in recently available Blend 3 build ? we call this feature ?design data?. This feature allows developers to enable designers to be more productive inside Blend for practical LOB applications where there is a clear separation between model and view, and where you can?t use the Blend sample data feature as your UI might depend on your custom business objects (for example, think typed data templates in WPF).

[Design Data Sample](#)

Let us take a quick tour of this example, and see how things work behind the covers:

ShoppingCart.cs defines a data structure that we are trying to visualize in ShoppingCartView.xaml. The type ShoppingCart does not have a public constructor, neither does ShoppingCartItem. Similarly, ShoppingCart has a read-only property ItemsCount. These kinds of limitations are extremely common in real-world data structures (and is being demonstrated here to show how our system handles these limitations just fine, without requiring your to make changes to your run-time code)

The sample data for ShoppingCartView.xaml is defined in ShoppingCartSampleData.xaml, which is included with a special build item type in the project file: . This ensures that you don't pay any run-time penalties for enabling this design-time only feature.

The contents of ShoppingCartSampleData.xaml look like following:

There are a few things to note about the contents of this file:

- a) While the format is XAML, we have relaxed the specification quite a bit. You are allowed to specify a

value for the read only property (ShoppingCart.ItemCount, ShoppingCartItem.ItemName), as well as initialize objects that don't have public constructors (ShoppingCart, ShoppingCartItem). Blend creates on the fly types which look like user types. You could choose to prevent reflection (in which case, what you can specify in XAML is severely restricted) by specifying true in the project file for the sample data XAML item.

- b) You can specify any platform type like Brushes, ImageSource, Uri, etc.
- c) You get full intellisense in Blend for typing this XAML.

In ShoppingCartView.xaml, we then hookup the sample data like so:

For WPF projects, if your XAML was using typed data templates, you need to add the following attribute to those typed data templates to get picked up automatically, if you were using the reflection based sample data types and you wanted a good design experience.

This particular sample data feature will also be supported in VS 2010. As always, please don't hesitate to ask for further clarifications, and I really value your feedback on how we could make this feature more useful for you.

<http://blogs.msdn.com/unnir/archive/2009/07/12/introducing-sample-data-for-developers.aspx>

Blend 3 Databinding

Here is a collection of a few of the blog posts I came across around the new databinding features in Blend 3. Hope you find them useful!

Sample data

<http://blogs.msdn.com/usisvde/archive/2009/06/17/blend-3-great-feature-1-sample-data.aspx>

<http://silverzine.com/tutorials/how-to-create-sample-data-in-blend-3/>

<http://www.85turns.com/2009/07/12/overview-of-sample-data-in-blend-3/>

[http://www.cynergysystems.com/blogs/page/michaelwolf?entry=silverlight blend 3 sample data](http://www.cynergysystems.com/blogs/page/michaelwolf?entry=silverlight+blend+3+sample+data)

<http://blogs.msdn.com/unnir/archive/2009/03/18/introducing-sample-data-support-in-blend-3.aspx>

Master/Detail scenarios

<http://blogs.msdn.com/usisvde/archive/2009/06/18/blend-3-great-feature-2-master-details-screens.aspx>

Element to Element binding

<http://blogs.msdn.com/usisvde/archive/2009/06/19/blend-3-great-feature-3-silverlight-element-binding.aspx>

TreeView control

<http://geekswithblogs.net/tkokke/archive/2009/06/29/styling-a-treeview-in-silverlight-and-blend-3.aspx>
<http://shawnoster.com/Blog/Silverlight-TreeView-Connecting-Lines-And-Blend-3-Support-for-HierarchicalDataTemplates>

DataGrid control

<http://blogs.msdn.com/unnir/archive/2009/03/19/datagrid-support-in-blend-3.aspx>

AutoCompleteBox control

<http://blogs.veracitysolutions.com/how-to-get-a-silverlight-3-autocompletebox-to-show-sample-data-in-blend-3/>

Under the covers (Advanced topics explaining the magic behind all this!)

<http://blogs.msdn.com/unnir/archive/2009/07/12/introducing-sample-data-for-developers.aspx>

<http://etvorun.spaces.live.com/>

<http://www.robfe.com/2009/08/design-time-data-in-expression-blend-3/>

<http://blogs.msdn.com/unnir/archive/2009/06/18/blend-3-databinding.aspx>

TabletPC and the Blend menus

If you do run into a situation where the Blend flyout menus appear on the left instead of the right as shown below, the Tablet PC Settings panel is a good place to check.

<http://blogs.msdn.com/unnir/archive/2009/06/07/tabletpcs-and-the-blend-menus.aspx>

Connect and Blend 3

While Blend 3 has a number of stellar features that I am sure everyone reading this blog has heard about (buzz words include SketchFlow, behaviors, sample data support, Illustrator and Photoshop import, TFS support, and many more), we really devoted a significant amount of time to address issues that you reported via Connect, that we hope will help you be more productive inside Blend. Here is a compilation of my favorite top-10 feedback items we addressed (in random order). Thank you very much to all you Connect contributors, and we really hope to hear more from you.

a) [Numeric editors for Gradient Stops](#)

Designers like to precisely control the positioning of gradient stops. Blend 3 adds this ability.

b) [Better handling of Layout properties in common operations like copy/paste](#)

With Blend 3, we have made a good, sincere attempt to streamline the properties that we set in common operations like copy/paste and double-click to insert. We also try to generate a clean as XAML as possible.

c) Consolidate the style and template editing menus

I could not find the Connect bug link for this one, sorry! While the concept of Styles and Templates offers enormous potential, it also make it challenging for a tool like Blend to offer access to them, and at the same time make us approachable to users who don't care about the separation. While we did a pretty good job of trying to abstract away the concepts of styles and templates, and while a lot more can be done, Blend 3 makes it possible to access all styles in the context menu. No need to go the Object menu to edit the ItemContainerStyle of a ListBox.

d) [AlternateContentProperty](#)

You can read about this in a previous post of mine [here](#).

e) [Expand/Collapse state of projects across solutions](#)

Each time you close and reopen a solution that contains multiple projects, we now remember the expansion states of each project in the solution. We also remember the set of documents you had open inside Blend so you can restart working where you left off immediately.

f) [Preserving XAML readability when trying to copy templates in WPF](#)

This is a WPF only improvement we made to Blend 3. In WPF, Blend allows you to edit copies of controls? templates by reverse engineering the XAML from the BAMLized resources. While this is great to have, it is also essentially lossy operation. As an example, any static resource references like Brushes get inlined (and consequently, the XAML becomes a little less user-friendly). Blend now allows custom control vendors to specify the original XAML file that should be used at design-time for template editing in the design-time assemblies (another post is required to show how you could take advantage of this if you were in the business of writing complex WPF controls and wanted to better enable designers). We also correctly copy the templates from the original Generic.xaml in many more cases when the original source is available.

g) [Typing in the split view XAML editor steals focus](#)

There were many cases where the design surface would steal focus from the XAML editor that would make it very annoying to type in the XAML editor. All of those should now be fixed, and along with support for intellisense in the XAML editor and a highly performant, robust-to-errors/exceptions XAML parser, typing in the XAML editor should be a really enjoyable experience for you (though we also work pretty hard to keep your XAML typing skills usage to a bare minimumJ).

h) [Arrange-by \(z-index\) is not preserved across sessions](#)

The order in which elements appear in XAML does not always play well with the cognitive understanding of who like to see elements that appears higher in z-order to appear higher in the object tree. While Blend 2 had the ability to switch the visualization order in the object tree, we made it hard for users who would have to set this every time they opened a solution. Not anymore!

i) [Keyboard shortcuts fine tuning](#)

This is a pretty hard problem to solve without free-form keyboard customizability and presets for shortcuts (a feature that is missing from Blend 3). But we made a pretty sincere effort here. For example, you can now customize the mouse zoom/scroll behavior for the design surface to your liking.

j) And finally? [Blend ? so bad](#)

We totally fixed this! Hope you enjoy using Blend 3 J.

<http://blogs.msdn.com/unnir/archive/2009/05/25/connect-and-blend-3.aspx>

The Blend 3 Asset Library

For Blend 3, we have completely re-designed the Asset Library.

Here are some of the highlights:

- a) Categorization for various assets makes discoverability easier.
- b) Searchability allows for quick location of an asset across categories like Controls, Effects and Behaviors.
- b) Freely dockable anywhere in the UI. We also have left the popup mode unchanged for quick one-time access to assets.
- c) Extensible - you can register your own assets that make it easy for inclusion into the projects on an on-demand basis. Adding an asset like a Control or a Behavior will add all necessary references required for the functioning of the project automatically.
- d) List and Grid modes for the display of assets.

Registering your own library (or sets of libraries) in the asset tool is very simple. All you have to do is to setup a registry key that points to a folder containing the libraries. As an example, the Silverlight SDK registers itself into the Blend asset tool using the following key/value pair on a 64-bit machine:

Key:

HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\Expression\Blend\3.0\Toolbox\Silverlight\v3.0\Silverlight SDK Client Libraries

Value: c:\Program Files (x86)\Microsoft SDKs\Silverlight\v3.0\Libraries\Client\

For a WPF example, the following is the way the WPF Toolkit registers itself on a 32-bit machine:

Key: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Expression\Blend\3.0\Toolbox\WPF\v3.0\WPFToolkit

Value: C:\Program Files\WPF Toolkit\v3.5.40320.1

There are a few things you can customize around the display of assets:

The Icon: For supplying a custom icon, all you need to do is to add an image to the control library (preferably its design-time library that helps keep the size of the library small) that uses the namespace qualified name of the control for its name. For example, say you had a custom control Foo. The icon would be then called FoosNamespace.Foo.png. Few things to keep in mind: Use EmbeddedResource as the build item for the Image. Blend's asset library only supports PNGs. If you wanted a 24x24 and 12x12 version of icons (since we use these two standard sizes in various places of our UI), all you have to do is to name the PNGs as follows: FoosNamespace.Foo.SmallIcon.PNG, and FoosNamespace.Foo.LargeIcon.PNG. The SmallIcon/LargeIcon part don't actually matter - you can pick a string of your choice, and we will dynamically determine and pick up the appropriate icon for display.

Description: You can use the DescriptionAttribute to change the string that is displayed as a tooltip for the asset.

Asset Library availability: If you wanted to prevent a particular asset from being visible in the Asset Library, you could use the ToolBoxBrowsable attribute as a part of the metadata specification for that asset in the design-time library.

Location in the category hierarchy: Again done via a newly introduced attribute (ToolBoxCategoryAttribute) that you can supply via a design-time library.

<http://blogs.msdn.com/unnir/archive/2009/05/22/the-blend-3-asset-library.aspx>

Marching towards the Blend 3 Beta

The last few weeks (or months) have been really hectic! We finally finished all the features that we wanted to add to Blend 3, and are now actively fixing all the feedback (read bugs and crashes :) that you have been reporting to us, as well as making performance tweaks to the product to make things work really well end-to-end. Please keep the feedback coming - we really appreciate it.

Here is a (random, off the top of my head) sneak peak into some of the new features we are adding to Blend 3 in the upcoming release, since the last Blend 3 Preview:

A completely re-designed Asset Library: This is by far the biggest addition to Blend, and I am sure you will love it - after all, its dockable! More information on this later. The sample data system now allows you to specify custom objects (like Brushes and your business objects), that just completely opens up the Blend sample data system for developers to better enable designers. I will post a sample of this shortly. In addition to this, we also added support for TreeView and binding to recursive collections. **Project pane search** - you can easily find the files in your project, similar to what you would do for properties. **Find in Files** - this allows you to search for information across files. Better paste functionality - we now allow you to **paste images from the clipboard**. You no longer need to build your projects to preview changes to UserControls. **Marquee selection of keyframes** that allows you to select multiple keyframes easily in the animation window. A lot more functionality to make **Behaviors** even more easy to create and use like an easier way to pick elements, re-use the element, storyboard and state pickers, etc. This is addition to newer set of Behaviors and Actions that will be available out-of-the-box. State editing experience - we now allow you to view **previews of transitions** on the design surface as you switch between states. **FluidLayout** is a new feature that we are adding that allows you to design animated, dynamic layout experiences. Support for **out-of-browser Silverlight applications**. A much improved, highly performant, **XAML editing experience** that is robust to the typing experience (i.e. a design surface does not throw exceptions or blanks out as you type in the XAML, and one that allows you to work with an element even when an exception is thrown so you could correct your errors in the UI). A Blend SDK that allows for Blend created projects to be portable between designers and developers without actually requiring a Blend install.

In addition, we also spent a significant amount of time polishing up the Blend UI. Just simple things that required attention to detail and will make your overall experience with the product enjoyable - hope you will like them.

Watch out [Christian's blog](#) for the SketchFlow additions - I think SketchFlow is clearly the crown jewel of Blend 3. We also spent a significant amount of time polishing up the Blend UI, and hope you will like the improvements.

Really looking forward to June and your feedback!

<http://blogs.msdn.com/unnir/archive/2009/05/22/marching-towards-the-blend-3-beta.aspx>

Blend, WPF and resource references

A number of people have sought more clarity around how Blend and resources references work inside WPF projects. Hope the following FAQs would help with some of your questions.

a) Should I use Static or Dynamic resource lookup?

Blend def. plays better with dynamic resource lookups. You could use a static resource lookup as long as the resource was not located or merged into App.xaml. People have raised concerns around performance issues with dynamic resource lookups (you pay for what you get). While that might be true, an interesting data point is that the Expression Blend source code uses a ton uses dynamic resource lookups for our own UI (of course, we too use static resource lookups in places where the resource would never change, or where it not possible to use a dynamic resource extension, for example non-DPs).

b) If I had to used a static resource lookup, why don't things work when the resource is located in App.xaml?

When a static resource lookup is done inside the Blend design surface, unfortunately, the resource ends up being looked inside the Application object of Blend (which is in itself a WPF application). This does not play well with the way we host the design surface - we don't want to merge the user resources into the Blend Application object to avoid conflicts with the Blend UI styles. Please be assured that solving this problem is pretty high our wish list (and we are working with the WPF team on a solution).

P.S.: If you do run into this issue, the most common symptom is that an exception is thrown on the design surface when you try to instantiate your control which reads something like - XamlParseException - Cannot find resource named '{Blah}'. Resource names are case sensitive...."

c) How can I create a control library of resources and use those resources in a different project?

[Here](#) is an example solution that shows you the setup.

d) What is a recommended pattern for organization of resources into external resource dictionaries?

Where there is no single pattern that I have noticed and it really depends on your scenario, an interesting data point is that the Blend source code itself only consists of a few resource dictionaries - we have one for a bunch of colors and brushes, and one for the styles for the common controls we use in our UI. Of course, we have two sets of these - one each for the Expression Dark and Light themes. Of course, fewer number of resource dictionaries helps with better performance inside Expression Blend.

e) What if I was instantiating resources from code? How can I make my application more design-time friendly?

[Here](#) is a blog post that might help.

<http://blogs.msdn.com/unnir/archive/2009/03/31/blend-wpf-and-resource-references.aspx>

AlternateContentProperty attribute

A new extension point we have added to Blend 3 (based on very popular demand) can be seen put to good use with the [Silverlight Chart control](#) - we call it the AlternateContentProperty attribute. Annotating the properties of a control with this attribute allows you to select objects that are set as values (including DependencyObjects) to be selected in the Blend object tree, thereby exposing a ton of new functionality like the ability to set properties on those objects, edit templates if those objects were FrameworkElements, or set new objects as values of these properties.

Kudos to the Silverlight toolkit team for jumping onto this (I hope a lot more people notice and use this feature

in Blend 3 as it really, really enables designers).

<http://blogs.msdn.com/unnir/archive/2009/03/28/alternatecontentproperty-attribute.aspx>

A sample that shows the new Blend design surface extensibility

Based on popular demand, [here](#) is a quick sample (WPF only though it would be very easy to port this as-is to Silverlight - currently busy with Blend 3 work and all the cool newer features we are adding to Blend 3) that will allow you to get started on the following new extension points we have added to Blend 3:

- a) The new way to specifying metadata for your controls
- b) DefaultInitializer that allows you to set properties when a control is instantiated from the Blend asset library
- c) Custom context menus
- d) An adorer for the control

Let me know if you wanted some samples for any specific scenarios you might be interested in.

<http://blogs.msdn.com/unnir/archive/2009/03/27/a-sample-that-shows-the-new-blend-design-surface-extensibility.aspx>

Writing a design time experience for a Silverlight control

Writing a design time experience for a Silverlight control can be a bit intimidating (the experience for WPF controls remains unchanged from Blend 2 - just that you now have a ton more extensibility points - unless you wanted to re-use the same design-time code for the WPF and Silverlight versions of your controls). Let's try and understand how this works (because once you know the basics, its a breeze!).

[Here](#) is a project template that will help you get started immediately. All you need to do to use this project template is to unzip the contents to a folder like C:\Users\Documents\Expression\Blend 3\ProjectTemplates. Also, sorry - I did not have time to create a VB version yet.

The project created from this project template works as follows:

Two control libraries are created each time you create and instance of the project template. For example, lets assume the the user chose the name "MySilverlightControl" as the name for the control library. The two libraries that are created are called MySilverlightControl.dll, and MySilverlightControl.Design.dll. The output path of MySilverlightControl.Design.csproj is set to copy the library alongside MySilverlightControl.dll in a sub-folder called Design. This is identical to what you would do for a WPF control library - adding the design time library to a sub-folder prevents pollution of the "Add Reference" dialogs in the tools. MySilverlightControl.Design.csproj is basically a WPF control library project (all the user interface components for the design time experience and metadata that you will specify for the controls are using the desktop CLR/WPF), and has the following: A project-to-project reference to MySilverlightControl.csproj Because MySilverlightControl.csproj is a Silverlight project (and because we want to use Silverlight types when we code the design time experience - more on this shortly), we need to add a reference to the Silverlight System.Windows.dll References to the two Blend 3 / VS Next shared extensibility libraries - Microsoft.Windows.Design.Interaction.dll, and Microsoft.Windows.Design.Extensibility.dll. (**Note:** In Blend 2/VS 2008, we also had Microsoft.Windows.Design.dll - the APIs in this dll have been moved into the other two libraries, and MWD.dll is no more) Because you have references to PresentationFramework.dll and System.Windows.dll, there is a conflict of types when you use a type like Button which exists in both. To help resolve this conflict for the compiler, we setup an alias for the Silverlight System.Windows.dll - TargetPlatform. Any type that is referenced using TargetPlatform::X.Y.Z is now resolved against the Silverlight assembly. For both WPF and Silverlight control libraries, while the way you specify the metadata tables is still compatible with VS 2008 / Blend 2, the registration mechanism has a breaking change. You now need to use an assembly level attribute - ProvideMetadata - for this. You can find an example for this in MetadataStore.cs in the MySilverlightControl.Design.csproj.

While there are clear benefits to the approach we have chosen to support both WPF and Silverlight controls (same extensibility APIs so you don't have to learn two sets of APIs, ability to reuse the code for your design time experiences for WPF and Silverlight version of your controls by simply creating platform specific versions of the design time libraries), there are some pitfalls for advanced scenarios that can be easily avoided:

There are breaking changes to the APIs. Because Visual Studio 2008 SP1 GACs assemblies, unless you had a strong reference to the new assemblies, you might get compiler errors if the assemblies don't resolve to the new versions. Another option is to copy the assemblies locally into the project, and reference them from there. This issue will be addressed in a more convenient fashion shortly. It is very easy to to unknowingly pass in a platform specific object (say System.Windows.Point) into an API (for example, the ModelItem ones to set values) where the platform is not compatible. This can be easily avoided by using aliasing appropriately. The .Net 3.5 SP1 WPF markup compiler (which comes into play when you try to define WPF UI for your desing-time - for example, a custom category layout for the property inspector or an adorner) does not play well with the aliasing system. To avoid this, consider separating the UI for the design time into a separate project that is a WPF only and has no references to the Silverlight assemblies.

Enjoy, and please do let me know if you run into issues as you work with Silverlight design time experiences so we can address them.

DataGrid support in Blend 3

Blend 3 adds really cool support for the DataGrid control (for both Silverlight 3 and [WPF Toolkit](#)). Here is a quick demo to help you get started. Enjoy!

Create a new Silverlight 3 application and instantiate the DataGrid control from the asset library. (As a side, this also demonstrates the newly added support for having custom controls in the Blend asset library - instantiating a custom control adds the necessary assembly references to the project, in this case System.Windows.Controls.Data.dll for Silverlight)

Create a data source that has a collection, with one or more properties. (You can easily do this using the new [sample data features](#) in Blend).

Drag and drop the collection onto the DataGrid - its that simple! Blend will automatically generate the right kind of columns for the various properties in the collection. (**TIP:** You can select individual properties using Shift+Select if you did not want a column for each property in the collection. You can even add the columns individually by dragging and dropping each property onto the DataGrid. You can right click on the DataGrid to add new columns if you wanted to go that route.)

Select a DataGrid column (the one that corresponds to the Image column), right click on the column, and edit the CellTemplate. You can now edit this template just like you would any other template. (**TIP:** The properties of the DataGrid columns are also available for manipulation via the property grid).

<http://blogs.msdn.com/unnir/archive/2009/03/19/datagrid-support-in-blend-3.aspx>

Introducing sample data support in Blend 3

To help you walk thru some of support we have added for sample data inside Blend 3, I wrote a hands-on-lab that takes you thru the basic experience of building a simple master/detail visualization. While there are a lot more scenarios that are possible with the support for sample data in Blend 3 (some of which are not available in the public preview build, and I will in write in detail about over the next few days), you can get a first-hand experience here for the following:

A schema designer that allows you to quickly mockup a data source and supply values for the dataImproved drag/drop experience to create bindingsMaster/Detail support in Blend 3Sample data that can be enabled/disabled when the application is runningOptionally, if you are developer and are interested in knowing more about how things work behind the covers so you could adapt this for your scenarios, this can be a good starting point.

Download the hands-on-lab document [here](#) (because of the ISP that I use for hosting these files, you will have to rename this .zip file to .docx), and the starting project (to save you some time with the graphics, styling, and layout) [here](#).

Expression Blend 3 Preview is available!

I am sure you must have heard - [Expression Blend 3 Preview is here!](#) And I am back after a 2 year break from writing on this blog - if you were to use the product, you will realize what kept us busy :).

If I were to list the new features in Blend 3, it would take me a day to write. Instead, I am going to list just the set of features that I personally was responsible for:

- a) **The new databinding experience** - We have radically redone the databinding experience in Blend 3. Be sure to check out the newly introduced support for sample data, which is a huge enabler in design scenarios that previously required writing code, or were just not possible. We also have brought to the table an unmatched DataGrid editing experience, and added have support for easily creating Master/Detail visualizations (Did I mention that creating the "Hello World" of RIAs - an RSS reader - is only a couple clicks away in Silverlight 3 using Blend 3?). More on all this shortly...
- b) **XAML Intellisense** - easily the number one requested feature of Blend. Also, Blend now has C# and VisualBasic code editing.
- c) **TFS support** - read more [here](#)
- d) **Silverlight and WPF Extensibility** - read more [here](#)

A bunch of other small things that I will cover over the next few days (there are also a few more tricks that we are still holding up our sleeves, and unfortunately, I won't be able to share much information about them). You also can watch a quick video of me introducing Blend 3 [here](#) (it is very, very hard to do justice to most Blend 3 features in 20 minutes, let alone all of them!).

Hope you enjoy using Silverlight 3 (my personal three favorite SL 3 features - shaders, projection transforms, and support for out-of-browser apps - which are yours?) and Blend 3, and please don't hesitate to ask questions.

<http://blogs.msdn.com/unnir/archive/2009/03/17/expression-blend-3-preview-is-available.aspx>

Blend 3 Extensibility

Based on popular customer feedback, we have added a number of new extensibility points in Blend 3 (sorry, no support for plugins, yet, but [who needs an officially supported extensibility model anyway?](#) :))

Over the next few weeks, I will provide more information on these, including interesting use cases that are already starting to pop up. Here is a quick listing of what we currently support:

Project and item templates: We support the same formats as Visual Studio, with some very minor modifications to suite our user experience. User project and item templates can be copied into C:\users\username\Documents\Expression\Blend 3\ProjectTemplates and \ItemTemplates respectively. We do not support wizards yet. The Blend Asset Library Ability to register your custom controls. For example, to register a control library Foo.dll, all you need to do is to add the path to the control library as a value of the following registry key -
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Expression\Blend\v3.0\Toolbox\Silverlight\v3.0\MyCustomControlLibraryKey. If this were a WPF library, the key you would setup is

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Expression\Blend\v3.0\Toolbox\WPF\v3.5\MyCustomControlLibraryKeyAdding an asset adds necessary assemblies to your projectSupport for custom control iconsSetting custom properties on your control when they get instantiated - [DefaultInitializer](#).Design surface for custom controls Custom context menu commandsAdornersSelection/object manipulation APIsProperty grid extensibility - same support as Blend 2, with a few minor additions here and there

Some other notes:

Blend 3 will share the same extensibility model as the next version of Visual Studio.

As we live in a world where there is greater and greater parity between WPF and Silverlight, we wanted the same for these APIs wherein you could use the same design time code to target controls for both platforms. To accommodate this, we had to make some minor breaking changes to the VS 2008 APIs (hopefully the changes will be very straightforward to adapt to) - I will try to post a number of samples to help out with this. By and large, the APIs remain the same as documented [here](#) (the documentation will be updated to reflect the breaking changes at a later date).

The APIs are not final yet, and are subject to change. Hopefully, we can keep the changes to a minimum.

<http://blogs.msdn.com/unnir/archive/2009/03/17/blend-3-extensibility.aspx>

Team Foundation Server support in Blend 3

Expression Blend 3 Preview adds support for integration with Team Foundation Server, one of our top feature requests.

Some examples of the various integration points:

- a) Saving a file automatically checks it out
- b) Adding a new UserControl or assets to a project automatically adds them to source control
- c) Renaming or deleting files automatically renames or deletes items under source control
- d) Right clicking on an item that has been modified under source control allows you to submit that particular change
- e) View history, get latest versions of files or specific versions, undo changes, etc.

To enable source control for a solution open inside Blend, the solution must be bound and residing in a valid workspace on the client. You can refer to the Visual Studio documentation on how to setup a solution under TFS source control [here](#).

While you don't need to have Visual Studio 2008 installed on the machine to avail TFS support inside Blend, you do need to install [Visual Studio Team System 2008 Team Explorer](#), a free download. You also need to install [SP1 of Team Explorer](#), and a hotfix for Team Explorer SP1 that enables TFS support inside Blend - you can download that from [here](#).

Enjoy!

<http://blogs.msdn.com/unnir/archive/2009/03/17/team-foundation-server-support-in-blend-3.aspx>
