Xaml Image Converter Manual

Installation

Run the SetupXamlImageconverter.msi to install the xaml image converter. All files will be installed in the MSBuild extensions path, i.e. C:\Program Files\MSBuild\JohnsHope Software\SkinBuilder\2.0

If you want to install the sourcecode, install TortoiseHG and clone the repository you find on http://xamlimageconverter.codeplex.com/SourceControl/list/changesets

Image Converter Batch files

Normally you tell the converter what to do through a xml batch file with the extension sb.xaml.

Such a batch file could look like this:

```
<?xml version="1.0" encoding="utf-8" ?>
<sb:SkinBuilder
  xmlns:sb="http://schemas.johnshope.com/SkinBuilder/2011"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">

  <sb:Scene OutputPath="~/img">
    <sb:Xaml Source="Image.xaml"/>
    <sb:Snapshot Filename="Image.png" />

    <sb:Group Element="IconBar" OutputPath="~/img/icons">
      <sb:Snapshot Filename="bar.png" />
      <sb:Snapshot Element="rotatingIcon" Storyboard="rotation" Frames="20" Filename="animated.gif" />
    </sb:Group>

    <sb:Snapshot Left="0" Top="0" Width="14" Height="164" Dpi="600" Filename="page_header_lft.png" />
    <sb:Snapshot Left="14" Top="0" Width="748" Height="164" Dpi="600" Filename="page_header_ctr.png" />

    <sb:Set>
      <Copyright.Content>
        <Button>Chris Cavanagh &amp;amp; johnshope.com</Button>
      </Copyright.Content>
    </sb:Set>
    <sb:Snapshot Element="Copyright" Filename="copyright_bar_element.png" />
    <sb:Reset/>

    <sb:Set Copyright.Content="Johnshope Software" />
    <sb:Snapshot Element="Copyright" Filename="copyright_bar_element.png" />
    <sb:Reset/>
  </sb:Scene>

  <sb:Scene OutputPath="~/img">
    <sb:Xaml Type="MyNamespace.Document"/>
    <sb:Snapshot Cultures="en;en-GB;fr" Element="Page1" Filename="doc.pdf" Page="a4" /> <!—Page 1 -->
    <sb:Snapshot Cultures="en;en-GB;fr" Element="Page2" Filename="doc.pdf" Page="1cmx5cm" />< !—Page 2 -->
  </sb:Scene>
</sb:SkinBuilder>
```

If you import the MSBuild target into a VisualStudio Project file (see below), you get intellisense in a batch file.

For each xaml file you want to convert you must specify a Scene element. Each Scene contains a single Xaml Source, that can be a file, a class or direct xaml in the batch file.
If you reference a class you specify the type of the class in the Type attribute. Optionally you can also specify an assembly with the Assembly attribute, or you can specify the assembly in the type name in the Type attribute.

For each Scene you can define many Snapshots. You also can Group the Snapshots with a Group Element, where you can define a common OutputPath or a xaml Element, that all Snapshots should be children of. For the Snapshots there are the following attributes:

- Filename: The name and filetype of the image file to create. Supported types are png, gif, jpg, jpeg, bmp, tif, tiff, wdp, pdf, xps.
- Element: The xaml element to take a snapshot of.
- Left, Top, Right, Bottom, Width, Height: The image position relative to the Scene's root Element, the Group's Element or the Snapshot's Element of the snapshot to take.
- Dpi: The final dots per inch of the output for raster images.
- Storyboard & Frames: For animated GIF's. The storyboard chooses a xaml storyboard and the Frames specified in frames per second. In the current implementation it is not possible to create loops.
- Page: For pdf & xps documents. Specifies the page size, either a name as "a4" or "letter" or a size as "10cmx20cm" or "13inx3in". If you omit the page size, the document's size will be the snapshot's size.
- FitToPage: For pdf & xps documents. If you specify the argument FitToPage="true" the image will be resized so it fit's to the page.
- For pdf & xps documents you can create multiple pages by just creating multiple snapshots that refer to the same file.

You can dynamically change the content of the xaml though the Set element. With Set you can either specify an attribute that refers to an element.attribute, like Button1.Text, or you can specify a child element of the form <Element.Attribute> with child xaml elements you want to assign to that attribute. You can revert the changes made with Set with the Reset or Undo element.

You can create culture specific images if you use compiled localized xaml. You can specify the Cultures attribute where you can specify one or more comma or semicolon separated cultures. The xaml is then converted with those cultures, to images with names ending on image.culure.imagetype, i.e. with a filename of apple.png and a Culture attribute of "en;en-GB;de" the files apple.en.png, apple.en-GB.png and apple.de.png are generated.

Setting up a VisualStudio Project for using the Xaml Image Converter

If you have some Xaml files in a VisualStudio project that you want to convert to images on building, you can import the SkinBuilder MSBuild target into your project:

Right-click on the project and choose "Unload Project". Next right-click on the greyed out project again and choose "Edit YourProject.csproj" where YourProject.csproj is the name of your project file.

Now edit the xml project file. At the end of the xml project file paste the following line:

```
<Import Project="$(MSBuildExtensionsPath)\JohnsHope Software\SkinBuilder\2.0\SkinBuilder.targets" />
```

Save the project file. Again right click on the project and choose "Reload Project".

Now Import the johnshope.SkinBuilder.dll into the references of your project (This step is mandatory. It is needed for Intellisense support).
Now right-click on the project and choose Add -> New Item... and create a new Xaml Page, and give it an extension of .sb.xaml. Edit the Xaml Page, change the root element to <sb:SkinBuilder> and add an xml namespace:

```
xmlns:sb="http://schemas.johnshope.com/SkinBuilder/2011"
```

Clear all elements inside the root element. Now add the <sb:Scene> element with a <sb:Xaml> element for your xaml source and one or more <sb:Snapshot> elements for the images you want to create.

Now save the sb.xaml. Next click on the sb.xaml file in the Solution Eplorer and go to the properties window. Choose a Build Action of either SkinBuilderPreCompile or SkinBuilderPostCompile.
SkinBuilderPreCompile is for loose xaml images, that get converted before compilation, so you can use the generated images as a resource.
SkinBuilderPostCompile is for compiled xaml images, so the xaml files get compiled and coverted afterwards.
Now build your project, and the images will be converted each time you modified the xaml source or the sb.xaml batch file.

Setting up a ASP.NET Site to use the Xaml Image Converter Handler

In order to use the Xaml Image Converter Handler, copy the johnshope.SkinBuilder.dll to your bin folder (This is not necessary if you installed XamlConverterHandler on the target machine, because johnshope.SkibBuilder.dll will be installed to the Global Assembly Cache). Next include the handler in your web.config like this:

For IIS7:

```
<system.webServer>
  <handlers>
    <add name="XamlImages" verb="*" path="*.sb.xaml" preCondition="integratedMode"
      type="johnshope.SkinBuilder.XamlImageHandler, johnshope.SkinBuilder, Version=2.4.0.0, Culture=neutral,
PublicKeyToken=60c2ec984bc1bb45" />
    <add name="XamlJpgImages" verb="*" path="*.xaml.jpg" preCondition="integratedMode"
      type="johnshope.SkinBuilder.XamlImageHandler, johnshope.SkinBuilder, Version=2.4.0.0, Culture=neutral,
PublicKeyToken=60c2ec984bc1bb45" />
    <add name="XamlPngImages" verb="*" path="*.xaml.png" preCondition="integratedMode"
      type="johnshope.SkinBuilder.XamlImageHandler, johnshope.SkinBuilder, Version=2.4.0.0, Culture=neutral,
PublicKeyToken=60c2ec984bc1bb45" />
  </handlers>
</system.webServer>
```

Or IIS6:

```
<system.web>
  <httpHandlers>
    <add verb="*" path="*.sb.xaml, *.xaml.jpg, *.xaml.png" type="johnshope.SkinBuilder.XamlImageHandler,
johnshope.SkinBuilder, Version=2.4.0.0, Culture=neutral, PublicKeyToken=60c2ec984bc1bb45" />
  </httpHandlers>
  <pages controlRenderingCompatibilityVersion="3.5" clientIDMode="AutoID" />
</system.web>
```

The first entry is a xaml handler for a converter batch file. You can reference the images of such a batch file from your ASP.NET code like this:

If you have a batch file ~/img/MakeImages.sb.xaml, that's creating an image ~/img/dummy.png you would refer to the image as ~/img/MakeImages.sb.xaml?image=~/img/dummy.png

If your browser request an url like this the first time, the handler executes the batch file and generates all files. Subsequent calls will only create those files anew that are older that the batch file. The hander will also create a *.sb.xaml.log file with the details of the conversion.

The next entries are for direct conversions of loose xaml files. You must specify an entry for each image type you want to support. The image type is of the form *.xaml.??? where ??? is your image file type. For supported types see above under Snapshot attributes. Also, here a log file with the conversion details is created. You can refer to those images directly by their name, i.e. if you have a xaml file ~/img/apple.xaml you can refer to it as ~/img/apple.xaml.png.

Note that the current version of the ASP.NET handler does support 3D xaml objects only in IIS6 and not in IIS7. This is due to WPF not working correctly under IIS7, and is simply because WPF is not tested in server environments. SkinBuilder.exe can also be run as a Windows Service, so the ASP.NET Pages will use the Service to convert images, but unfortunately it still has the same issues as when converting under IIS7.

There are some further settings you can configure, in the web.config in the root configuration section:

```
<configSections>
  <section name="XamlImages" type="johnshope.SkinBuilder.XamlImageHandler"/>
</configSections>
<XamlImages Log="true" />
```

Here you can choose if the XamlImageHandler should generate a log file.

Command line version of the converter

In the MSBuild extensions path is also a command line version of the converter, though most of the time you will use msbuild with the msbuild tasks. The command line program is called SkinBuilder.exe. Note that the exe requires the johnshope.SkinBuilder.dll. For pdf support it requires also the gxps.exe in the gxps folder.

Contribute to the Code

If you're a Web Designer and Programmer and use Xaml Image Converter, and it lacks a feature you need don't hesitate to improve it. There are many useful possible extensions:

- ASP.NET ImageMap creation: Automatic creation of ImageMaps from shapes of Xaml elements. The converter could create an ascx control with the ImageMap. This would be very useful for complicated shapes.
- VisualStudio integration. Basically a Document Item Template for *.sb.xaml batch files, with a Template Wizard that creates the import statement for the msbuild targes in the project file.
- Automatic conversion of Expression Design, Photoshop & Illustrator files to Xaml and then to images.
- You can write an Email to Microsoft, that they should make WPF work under IIS for 3D xaml elements. Shouldn't be too hard for Microsoft. If you write the Email, there's also a bug in RenderTargetBitmap.Render when you render very many big bitmaps with 1200 DPI and the like, then the bitmaps don't get rendered completely. (I've created a color bitmap font that way).

Render does not wait correctly for the drawing thread to finish drawing the bitmaps. I have an example of this ready to reproduce the bug, if someone's interested.