EmbeddingSVGFonts

Introduction.

In the Cocoon world, often SVG is employed as an XML-friendly means of creating raster graphics through the SVGSerializer. Cocoon's use of SVG can, of course, be much more complex than that: with the Adobe SVG Viewer's (v3) support for animation elements, Cocoon can be used to generate SVG that rivals Flash in graphical user experience. For instance, for the last two years at the Historical Event Markup and Linking Project (Heml), we've been using Cocoon to generate SVG timelines and animated historical maps. (We can rasterize out static versions of these materials, but hyperlinking and animations are lost in the process.)

Because of its origins in teaching ancient Greek history, Heml aims to be highly multilingual. Using server-generated SVG in such a circumstance inevitably leads to frustration over fonts. A map with multi-lingual text carefully layed out using the Bitstream Cyberbit font will e riddled with 'missing character' squares when rendered on a client's machine that lacks Cyberbit.

The SVG specification does offer a solution to this problem: scalable font glyphs can be described within an svg:font element. With these, the client renders text using the appropriate SVG glyphs, not relying on font descriptions from the client's operating environment. But it would be completely impractical to render the entire, e.g., Cyberbit font as an SVG font and expect the client machine to upload this with each view. What is needed is a mechanism that appends to an SVG document only those glyphs necessary for it to be rendered. As it turns out, the marvellous Batik SVG Graphics2D environment has just this ability. I've tapped into it to make an xslt transformation (extended with xalan/java) which takes any SVG document and adds to it the glyphs necessary for its viewing. The result is a somewhat heavier document, but one that is completely portable.

Demo

This timeline of Russian history is laid out with a cursive font which probably isn't on your machine. You see the text in that font because the server automatically added its glyphs, resulting in a 21kb svgz document.

Use

The use the SVGFontEmbedder, you'll need to ensure that:

1. Your Cocoon environment has the Batik jar file. Check WEB-INF/libs for a file called something like batik-all.jar. (I've tested this with Cocoon 2.0.4, which comes with the 1.5b2 Batik code.)
2. You can process an xslt file with Xalan. Xalan is the default xslt processor in Cocoon, but some people have patched in support for Saxon because of its superior speed. (It should be noted that, at present, its speed comes at the cost of being able to pass off its elements to java processes as DOM objects, a deal-breaker for me.)

You'll also need to download two files:

1. `heml-cocoon-0_5.4-dev-20030221.jar` or later, found at http://heml.mta.ca/releases. You **don't** need the massive `heml-cocoon.war` file.
2. `svgEmbedFonts.xsl`, the latest version of which is easiest to aquire at its view cvs page

Install `heml.jar` in your $COCOON_HOME/WEB-INF/libs directory, and put svgEmbedFonts.xsl wherever your xsl files usually reside. Emend your sitemap to transform your SVG files just before they are serialized as svg+xml. (Incidentally, there is no point in using this transformation before rasterizing SVG to jpg or png, since your server will have access to its own fonts when rasterizing.) For instance, an appropriate part of my sitemap looks like this:

```
                <map:when test="map">
```

```
                  <map:transform src="xslt/heml/output/svg/map/dynamic_map.xsl">
```

```
                    <map:parameter name="context" value="{../../context}"/>
```

```
                    <map:parameter name="request_lang" value="{lang}"/>
```

```
                  </map:transform>
```

*

```
                  <map:transform src="xslt/util/svgEmbedFonts.xsl"/>
```

*

```
            </map:when>
```

Once Cocoon is duly restarted, etc., your SVG should now have the necessary `svg:glyph` elements automagically added.

Caveats

1. This code will only recognize fonts specified with the `font-family="MY_FONT"` attribute notation. You can't use `style="font-family: MY_FONT"` and expect this process to assign the right font to your text. On the positive side, the font-family can be specified on the `svg:text` element or any of its ancestors, just like in any honest SVG document. 1.It shouldn't need to be said that your Java environment has to have access to the font file in order for this code to work. This often means putting the FONT.ttf file in the $JAVA_HOME/jre/lib/fonts directory. Interestingly, if a glyph isn't defined, Batik squiggle will render it with some sort of default font, while ASV3 will give an 'unknown character' square.

Thanks

I feel I must emphazise that the Batik team provides 99% of the cleverness and hard work that makes this transformation happen. Kudos to them!

Additional Notes

The underlying java code has a javadoc page and can be downloaded with the rest of the Heml project at our CVS server.

Uncompressed, the files resulting from this transformation can get pretty large. I compress my SVG on the fly using a custom GzipXMLSerializer.

Ideas for Improvements

- Some industrious Cocooner might want to extend it into a cocoon transformer, thereby removing the need for the xslt file. I used the xslt because it gives me the power of xpath to pick up the font name from ancestors.
- Furthermore, someone who understands Batik 2DGraphics internal better than I might be able to grab the font defs more elegantly and reliably than by descending through the document tree from getRoot() as my code does.
- xpath or other mojo that could pick fontnames off of `style="font-family: MY_FONT"` types of declarations (see Caveats #2 above) would add greatly to the universality of this solution.

Futher discussion about this work is encouraged on the Heml listserv.