

# GzipXMLSerializer

## Introduction

XML is meant to be human-readable. Often this means that the markup is pointlessly verbose from the machine's standpoint. The common solution is to compress the XML – the SVG spec, for instance, stipulates that SVG viewers need to be able to read gzipped files as well as uncompressed equivalent; and numerous applications, from Gnumeric to OpenOffice, compress their XML before saving. When files are being sent over the internet, compression can make the difference between impracticality and practicality. Working with SVG maps at the [Heml Project](#), I found I was producing files well over 500k in size. These could never be served to the modem-connected great unwashed, but their gzipped equivalent, weighing in at less than 100k, could.

For more than a year, I used the mod\_gzip module for Apache to do the compression – an effort partly described at [Cocoon%and%Apache%mod%gzip --](#), but this was never entirely satisfactory. mod\_gzip is meant for compressing html and is based on the httpd negotiation between client and server. A client must declare in its request header that it can handle gzip content. However, SVG viewers and plugins rightly won't send this information, yet they can all digest svgz content. Secondly, any post-cocoon solution means that cocoon's cache will gobble up more of your RAM by caching the uncompressed XML. If we could serialize the result within Cocoon, its cache would hold the smaller gzipped version. (Someone correct me if Cocoon in fact compresses the contents of its cache.)

## About GzipXMLSerializer

After some fiddling, I have come up with a hack on Cocoon's XMLSerializer that serializes compressed XML content. For those interesting in the programming, the code is available through a [ViewCVS page](#). If this were to be integrated into Cocoon's API, I would recommend factoring out what is common between XMLSerializer and this, then making both a subclass. If you just want to use the darn thing to compress your own SVG, read on.

## Using GzipXMLSerializer

To use the GzipXMLSerializer, you'll need to:

Download `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. This should go in your `$TOMCAT/cocoon/WEB-INF/lib` directory.

Next, you'll need to declare the serializer in the Serializers section of your sitemap, thus:

```
<map:serializers default="html">
  <map:serializer
    name="gzip" mime-type="image/svg+xml"
    src="org.heml.cocoon.serialization.GzipXMLSerializer"/>
  <!-- all the other usual definitions go here -->
  <!-- ... -->
</map:serializers>
```

Finally, use the serializer as you would any other. I map the filename extension in this way so that we can get png, svg or svgz :

```
<!-- the part of the name after the dot indicates what sort of output we want -->
<map:select type="parameter">
  <map:parameter
    name="parameter-selector-test"
    value="{.../.../2}"/>
  <map:when test="svg">
    <map:serialize type="svgxml"/>
  </map:when>
  <map:when test="jpg">
    <map:serialize type="svg2jpeg"/>
  </map:when>
  <map:when test="png">
    <map:serialize type="svg2png"/>
  </map:when>
  <map:when test="rss">
    <map:serialize type="xml"/>
  </map:when>
  <map:when test="xml">
    <map:serialize type="xml"/>
  </map:when>
  <map:when test="svgz">
    <map:serialize type="gzip"/>
  </map:when>
</map:select>
```

Of course, if you wanted to use GzipXMLSerializer for other sorts of content, then you'll need to change the mime-type declaration on the serializer element.

## Examples

If you have Adobe SVG Viewer, you should get svgz content in:

- [A Timeline](#)
- [An Historical Map](#)
- [An Animated Historical Map](#)


## Performance

This approach uses Java's gzipping code to do the dirty work. You might think this would be less efficient than mod\_gzip, which uses some wicked-fast C, but I find no appreciable difference on my Athlon +1700 XP box.

## Related Material

I have written a Wiki page on an Xalan transformation that describes [EmbeddingSVGFonts](#) into a SVG document.

## Attached Files

The attached Java class resolves a bug  in the endDocument() method. The tranform handler needs to be notified of the end of the document event before any stream management is done.

**Attachment:** [GzipXMLSerializer.java](#)

## Comments

I used this serializer to compress a generated javascript files. IE and Firefox were not able to decompress the file. The HTTP response needs an header "Content-Encoding" set to "gzip". It is easy to add this header in an Action class.