FOP Design: Images

$Revision: 426576 $

Table of contents

1. Introduction

Images may only be needed to be loaded when the image is rendered to the output or to find the dimensions.
An image url may be invalid, this can be costly to find out so we need to keep a list of invalid image urls.

We have a number of different caching schemes that are possible.

All images are referred to using the url given in the XSL:FO after removing "url(")" wrapping. This does not include any sort of resolving such as relative -> absolute. The external graphic in the FO Tree and the image area in the Area Tree only have the url as a reference. The images are handled through a static interface in ImageFactory.

2. Threading

In a single threaded case with one document the image should be released as soon as the renderer caches it. If there are multiple documents then the images could be held in a weak cache in case another document needs to load the same image.

In a multi threaded case many threads could be attempting to get the same image. We need to make sure an image will only be loaded once at a particular time. Once a particular document is finished then we can move all the images to a common weak cache.

3. Caches

3.1. LRU

All images are in a common cache regardless of context. To limit the size of the cache the LRU image is removed to keep the amount of memory used low. Each image can supply the amount of data held in memory.

3.2. Context

Images are cached according to the context, using the FOUserAgent as a key. Once the context is finished the images are added to a common weak hashmap so that other contexts can load these images or the data will be garbage collected if required.

If images are to be used commonly then we cannot dispose of data in the FopImage when cached by the renderer. Also if different contexts have different base directories for resolving relative url's then the loading and caching must be separate. We can have a cache that shares images among all contexts or only loads an image for a context.

The cache uses an image loader so that it can synchronize the image loading on an image by image basis. Finding and adding an image loader to the cache is also synchronized to prevent thread problems.

4. Invalid Images

If an image cannot be loaded for some reason, for example the url is invalid or the image data is corrupt or an unknown type. Then it should only attempt to load the image once. All other attempts to get the image should return null so that it can be easily handled.
This will prevent any extra processing or waiting.

5. Reading

Once a stream is opened for the image url then a set of image readers is used to determine what type of image it is. The reader can peek at the image header or if necessary load the image. The reader can also get the image size at this stage. The reader then can provide the mime type to create the image object to load the rest of the information.

6. Data

The data usually need for an image is the size and either a bitmap or the original data. Images such as jpeg and eps can be embedded into the document with the original data. SVG images are converted into a DOM which needs to be rendered to the PDF. Other images such as gif, tiff etc. are converted into a bitmap. Data is loaded by the FopImage by calling load(type) where type is the type of data to load.

7. Rendering

Different renderers need to have the information in different forms.

7.1. PDF

original data
JPG, EPS
bitmap
gif, tiff, bmp, png
other
SVG

7.2. PS

bitmap
JPG, gif, tiff, bmp, png
other
SVG

7.3. awt

bitmap
JPG, gif, tiff, bmp, png
other
SVG

The renderer uses the url to retrieve the image from the ImageFactory and then load the required data depending on the image mime type. If the renderer can insert the image into the document and use that data for all future references of the same image then it can cache the reference in the renderer and the image can be released from the image cache.