INFO: HtmlDecoder Deep Dive

Our HtmlDecoder (New in 11.5) can render HTML files as images.

This has been a long asked for feature.

Dependencies

- You need to be using DotImage version 11.5.0.0 or newer
- You need a license for Atalasoft DotImage Document Imaging (DID2/DIDX2 serial)
- You need to reference Atalasoft.dotImage.CommonDecoders.dll
- You will need to ensure you add the Perceptive Filter Dlls (Shipped with DotImage) to your app bin folder (see below)

Perceptive Filters

The perceptive filters "have bitness" like the rest of DotImage.. so you need to pick which you're targeting (x86 or x64) and use the files from the appropriate source folder:

C:\Program Files (x86)\Atalasoft\DotImage 11.5\bin\PerceptiveDocumentFilters\intel-32

C:\Program Files (x86)\Atalasoft\DotImage 11.5\bin\PerceptiveDocumentFilters\intel-64

You can just copy the DLLs to your app bin folder or you can add to your project and set their build action to "Copy if newer"

So long as the 4 files end up in your final bin directory next to Atalasoft.dotImage.CommonDecoders.dll

Using the Decoder

Many long time DotImage users will probably assume the next step would be to add a new HtmlDecoder to RegisteredDecoders.Decoders collection in a static constructor.

The short answer is yes you can, but if you do, then you will find that HTML with img src tags that point to local files (such as <img src="foo.png">) will not render in the output HTML. However, those that reference web-reachable (by the program doing the decoding)

images will (such as <img src="https://www.atalasupport.net/images/logo_atalasoft_from_tungsten.jpeg">)

So, if you know your HTML files will never need to render local / relative images, this may be OK for you.

If you do need to render images, see the section below named "Known Issue: img src local image files not rendered"

HtmlDecoder Properties

Defaults

The defaults of new HtmlDecoder are

- PageSize: Size = {0 , 0} (when set to 0,0,the decoder will end up outputting a US Letter size page at Resolution - lots to unpack see below)
- Resolution: 96
- RenderExternalImages: true

Explanations

RenderExternalImages

This setting tells the decoder whether it is allowed to make web requests so that it can render image tags that point to web-reachable images.

If this is set to true, then image tags that refer to fully qualified URLs will be parsed and the decoder will attempt to fetch the image to use in rendering. For this to work, the program running the decoder must be able to reach the URL in the image tag - it will work for HTTP or HTTPS requests, but the url must be reachable without further authentication

If this is set to false, then no external requests will be made and the default "image not found" rendering wil be used

INFO: HtmlDecoder Deep Dive

PageSize

This setting tells the decoder what the target page size for the image render is...

IMPORTANT NOTE ABOUT PAGE SIZE:

Page Size value here is in points... meaning 1/72 of an inch. This may be unintuitive at first because if let 0,0 you will get a US letter page at whatever Resolution was set

for the default Resolution of 96, this would be Size = {Width = 816 Height = 1056}

Except if you plug new Size(816,1056) into PageSize and see Resolution 96, you might expect a output image with {Width = 816 Height = 1056} but you;ll actually get an output image that is Size = {Width = 1088 Height = 1408}

Agian this is because for the PageSize it's looking for POINT size of a virtual physical page you'd be looking to fit the object ont.

So,

PageSize = new Size(RealPhysicalPageWidth_in_Inches * 72,
RealPhysicalPageHeight_in_Inches * 72);

- Us Letter (8.5x11) would be 612,792
- Us Letter Landscape (11x8.5) would be 792,612
- US Legal Portrait (8.5x14) would be 612, 1008)

NOTE that the Decoder will hard cut the page width, dropping content to the right.. if your horizontal content is getting cut off, make the width wider... Meanwhile the decoder will make a new page at the height boundary and make as many pages as needed

Resolution

This setting tells the decoder how to interpret the image.

The formula to calculate the output Image size is

ImageOutputSize = { Width = PageSize.Width / 72 * Resolution, Height= PageSize.Height / 72 * Resolution }

So, for simple sake, lets say PageSize is the default of 612 x 792 (US Letter):

- Default Resolution: (96) the image will be {Width = 816 Height = 1056}
- 200 DPI: the image will be {Width = 1700 Height = 2200 }
- 300 DPI: the image will be {Width = 2550 Height = 3300 }

Combining resolution and Page size and considering that the control will paginate vertically but not horizontally, the suggested settings are

HtmlDecoder htmlDecoder = new HtmlDecoder() { PageSize = new Size(1008,612), Resolution = 200 };

This will result in 2800 x 1700 pixel images which unless your content is super wide should fit well

Tweak the PageSize to widen or narrow for content, tweak the Resolution to make the output image have higher quality

Known Issue: img src local image files not rendered

However, HTML with <img src="local file here"> src attributes do not render.

The decoder has a RenderExternalImages property which if set to true will properly render reachable images (the decoder needs to make a web request and reach the url containing the image at the time of rendering) - set this to false to prevent any external requests at the cost of un-rendered images

However, if your HTML has img src tags pointing at local resolvable files, you may find that the images do not render properly if you use

AtalaImage img = new AtalaImage("htmlfile.htm");

or in an image source

or even

AtalaImage img = htmlDecoder.Read(stream ...)

This is because of a limitation with the Perceptive Filter used to render HTML.

The workaround is that it will render local image tags (where the path can find the image in question) if and only if you use the HtmlDecoder.Read overload that takes the string filename as the first arg

AtalaImage img = htmlDecoder.Read("C:\\path-to\\example.htm", frameIndex, null);

HtmlImageSource

An ImageSource for HTML

Since this is not entirely intuitive and it's a common use case to want to render html as PDF pages, and the PdfEncoder requires a "one shot decoder" and works best with ImageSource, here is a HtmlImageSource that handles the issue fairly well

Attached to this article, find HtmlImageSouce.zip

It contains an HtmlImageSource class you can use

usage:

Make a PDF from an HTML file with 14"x8.5" pages

```
sing (AtalasoftExamples.HtmlImageSource his = new
AtalasoftExamples.HtmlImageSource("test.htm", new Size(1008,612), 200) { using (FileStream
outStream = new FileStream("out.pdf", FileMode.Create)) { PdfEncoder pdfEnc = new
PdfEncoder(new Size(1088, 200)); pdfEnc.SizeMode = PdfPageSizeMode.FitToPage;
pdfEnc.Save(outStream, his, null); } }
```

INFO: HtmlDecoder Deep Dive

Atalasoft Knowledge Base
https://www.atalasoft.com/kb2/KB/50442/INFO-HtmlDecoder-Deep-Dive