FIX: PdfImageSource Does Not Properly Handle Image Resolution

The PdfImageSource is a a highly useful component that allows developers to attempt to extract images embedded in some PDFs instead of rasterizing. This has a benefit of potentially ending up with smaller file sizes and less lossy interaction with decoding

The down side is that Images embedded within a PDF have their resolution information lost (Resolution is not part of the internals of a PDF), So, when using a PdfImageSource the resolution of the extracted embedded images will report 96 DPI regardless of what the original really was.

This Custom image source gets around that by reverse engineering the resolution by examining the pdf page size (inches) and using the known fact that those sizes are fixed at 72 DPI to reverse engineer the effective resolution for the image

C#:

```
sing System; using System.IO; using Atalasoft.Imaging; using Atalasoft.Imaging.ImageSources; using System.Drawing; using System.Collections.Generic; using Atalasoft.Imaging.Codec.Pdf; namespace AtalasoftSupportUtils { // in this class, we are subclassing PdfImageSource and adding some post processing to properly calculate the image resolution // this is needed because the PdfImageSource fails to properly report the resolution of extracted files // public class PdfAccurateResolutionFixImageSource : PdfImageSource { private List _sizes = new List(); public PdfAccurateResolutionFixImageSource(Stream stm) : base(stm) { PopulateSizes(stm, null); } public PdfAccurateResolutionFixImageSource(Stream stm, string password) : base(stm, password) { PopulateSizes(stm, password); } // this method overrides the LowLevelAcquire Method in PdfImageSource // It lets PdfImageSource get the image (possibly extracting it directly if it's a single // image page, then we calculate the image resolution and also // try to cut the image bit from 32Bpp to 24 if applicable protected override ImageSourceNode LowLevelAcquire(int index) { ImageSourceNode node = base.LowLevelAcquire(index); if (node == null) return null; node.Image.Resolution = CalcResValue(node.Image.Size.Width, node.Image.Size.Height, this._sizes[node.Index]); // ALSO taking the trouble to reduce image from 32bppBgra to 24bppBgr to make for smaller ouput AtalaImage image = null; if (node.Image.PixelFormat == PixelFormat.Pixel32bppBgra || node.Image.PixelFormat == PixelFormat.Pixel32bppBgr || node.Image.PixelFormat == PixelFormat.Pixel48bppBgr) { image = node.Image.GetChangedPixelFormat(PixelFormat.Pixel24bppBgr); } // couldn't reduce if (image == null) return node; // don't need original image anymore node.Image.Dispose(); node.Dispose(); //disposes node.Reloader. By some reason, this method doesn't dispose the node.image too. return new ImageSourceNode(image, new FileReloader(image)); } private Dpi CalcResValue(double imgWidth, double imgHeight, SizeF pgSize) { return new Dpi((72 * imgWidth / pgSize.Width),(72 * imgHeight / pgSize.Height), ResolutionUnit.DotsPerInch); } private void PopulateSizes(Stream stm, string password) { stm.Seek(0, SeekOrigin.Begin); Document d = new Document(stm, password); foreach (Page p in d.Pages) { this._sizes.Add(new SizeF((float)p.Width, (float)p.Height)); } d.Dispose(); stm.Seek(0, SeekOrigin.Begin); } } }
```

FIX: PdfImageSource Does Not Properly Handle Image Resolution

Atalasoft Knowledge Base
https://www.atalasoft.com/kb2/KB/50397/FIX-PdfImageSource-Does-Not-Properly...