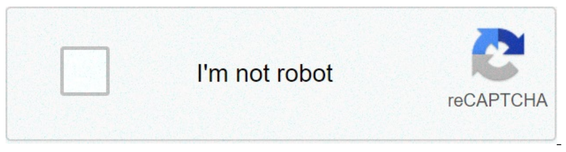

Continue

Itext add footer to existing pdf

So… This post spun a little out of control. What turned into a hunt for a simple use case (Turn an HTML string into a PDF) turned into a full blown thesis. So here’s some quick links to jump to various parts of the article if you are coming here straight from Google, it may make things a bit easier. But I do highly recommend reading from the start to get an idea of what PDF Generation really looks like on .NET Core.Our sponsor, the DocRaptor HTML-to PDF API, lets you create complex PDFs from HTML, CSS, and JavaScript. Only DocRaptor’s API supports advanced features such as varying headers and footers, page break fine-tuning, accessible PDFs, and more. Pricing starts at just $15/mo.IntroductionIt’s a pretty common use case to want to generate PDF’s in C# code, either to serve directly to a user or to save locally. When I came to do it recently, suddenly I was over(and under)whelmed with the options available to actually achieve this. Certainly natively, there is nothing in C# or .NET Core that can generate PDF’s for you. And when it came to actually looking at feature sets of third party libraries (e.g. I want to use HTML as a template option), there were all sorts of problems.So with that being said, instead of giving you code to generate a PDF with a particular PDF library, I’ll talk about how I evaluated each option.What I Was Looking ForBefore I set out on my journey, I wrote down 3 really crucial points that I would judge the libraries on. These aren’t too complex or even that much of an ask (Or so I hoped).PriceObviously free is ideal. Something open source that I can debug myself is even better. But if I do have to pay for a “premium” library. I’m looking for a one time fee that isn’t some stupid “per user/seat/machine/server” model. If I’m looking at this library as a company, I don’t want future architecture or decisions to be made based on the pricing of a library.If there is some sort of freemium model in play, then I also wanted to make sure that the limitations weren’t too crazy (e.g. single pages only, set number of images allowed per PDF). Freemium is OK as long as the free version is actually useable.HTML Templating (Or something close)I had already decided that I wanted to use HTML as my templating mechanism. I was open to using some other reasonable alternative (e.g. HTML with some XSLT engine), but ideally I just want to feed an HTML file to the library and out comes my PDF. What I really don’t want to do is have to place each element manually on the PDF like we had to back in the day when printing a document from a WinForms application.Ease Of Use/All In OneThis is probably a pretty subjective one, but when you start seeking out libraries from the corners of the web or that stackoverflow answer from 3 years ago, you often end up getting some really half baked library that just doesn’t work. Whether it’s some C++ library converted to C#, a library that needs X number of other libraries to actually function, or things just plain don’t work how they should, I’m probably going to see it all. So above else, I just want to be up and running in minutes, not hours.PDF SharpFirst up is PDF Sharp, I feel like I’ve used this one previously but I’m not entirely sure. The whole ___Sharp thing was all the rage in the early days of C#. Anyway straight off the bat PDFSharp does not work with .NET Core. There may be ported versions floating around but the version on Nuget does not support .NET Core. So it’s pretty much dead in the water. But this one was suggested to me over and over so I still want to do a quick write up about it.PricePDF Sharp is free and open source. They do offer paid support but don’t give any specifics on how this works and how much it costs. Free is free though so you can’t complain too much.HTML TemplatingOh boy. HTML Templating doesn’t make it into the PDF Sharp library unfortunately. Instead we are stuck with writing syntax like we are back in the GDI+ dark days of C# :// Get an XGraphics object for drawing XGraphics gfx = XGraphics.FromPdfPage(page); // Create a font XFont font = new XFont("Verdana", 20, XFontStyle.BoldItalic); // Draw the text gfx.DrawString("Hello, World!", font, XBrushes.Black, new XRect(0, 0, page.Width, page.Height), XStringFormats.Center);I get it that this was what you had to do in the year 2000, but it’s just not going to fly right now. There are ways around this using another library that extends PDFSharp… But that’s still another library that you have to grab and work with. So basically, in terms of a decent template engine out of the box, it’s a fail.Ease Of UseWell. It doesn’t work with .NET Core which is probably going to be a blocker. The last published version was 2013. And it doesn’t have HTML rendering packaged (And the library that currently does extend it to do HTML templating also doesn’t support .NET Core). So all in all. Forget about it.SelectPDFI got pushed to SelectPDF by a tonne of stackoverflow answers (Kinda felt like maybe they were astroturfing a little…). It’s a paid library (Which made it all the more annoying they were commenting on stackoverflow answers with “Sure just use this great library called SelectPDF” and not mentioning the cost), But regardless I wanted to check them out and see what they offered.PriceIt ain’t free, that’s for sure. Licensing starts at $499 and goes up to $1599. Interestingly they do offer an online version where they can generate the PDF for you by sending a URL or HTML code. Honestly I think that’s way more interesting as a business model, but the pricing again starts at $19 but goes all the way up to $449 a month so I can’t really see many people taking this option.EDIT : So weirdly enough, there seems to be a “hidden” free community edition that you can only find via a tiny footer link. You can read more about the Community Edition here. It seems like the only limitation is that you can’t generate PDFs over 5 pages. Doesn’t seem to be any limitation on commercial use. But again, seems strange to only make this a tiny footer link and not sure it anywhere else…HTML TemplatingHTML Templating is definitely here. And it seems to work well. In just a few lines we are up and running. While documentation is pretty good, I found it sort of all over the place. Loading HTML from a string versus loading HTML from a URL was all mixed together and so when reading examples, you had to check the code example to see what it was actually used for. But that being said, HTML Templating works so tick!Ease Of UseHere’s the code to render HTML :var myHtml = "h1 {font-size:12px;}TestTest Bold"; HtmlToPdf converter = new HtmlToPdf(); PdfDocument doc = converter.ConvertHtmlString(myHtml); doc.Save("output.pdf"); doc.Close();So on that level, ease of use seems good. I would say that many code examples were done using old school .NET WebForms which made me uneasy. Not sure why but using such an old technology like that doesn’t make me feel like I’m using the best in class.It’s hard to fault SelectPDF on the actual bones of what it’s doing. It generates PDF’s and does what it says on the tin. But I can’t help but feel like it’s a “budget” option for the price – It’s probably the logo not being transparent.IronPDFLast up, IronPDF. These guys create a bunch of libraries called IRON_____ so you may have run into them before. I’ve run into them a few times for their OCR library, and that’s unfortunate because as soon as they came up, I knew the pricing was gonna be out of this world.PriceFor starters they have all these tiers. Like you can get a single project license, but you have to pay per developer (ugh). But.. If you are a SAAS company, then you have to pay per user of your SAAS product. If you are developing desktop software or something distributed, well that’s a new license again. But let’s just stick with being a SAAS company for now, for 1000+ users I’m going to be paying $1599USD for the library and a years worth of support.In fairness, most large companies aren’t going to bat an eye at that. It’s not super crazy, and it is one off, but it’s still pretty pricey.HTML TemplatingYes and yes. In IronPDF it just works. What impressed me the most about IronPDF was their documentation. It wasn’t just “Here’s a snippet, good luck!”. They even went as far to show you examples of how you might use Handlebars as a templating engine, implementing page breaks in your HTML, and even watermarking. To say that it’s feature rich would be an understatement.Ease Of UseAgain, IronPDF comes up trumps when it comes to ease of use. In literally 4 lines everything just works and the output PDF is exactly as it should be :var myHtml = "h1 {font-size:12px;}TestTest Bold"; var htmlToPdf = new HtmlToPdf(); var pdf = htmlToPdf.RenderHtmlAsPdf(myHtml); pdf.SaveAs("output.pdf");Now here I’m using inline HTML, but I can load an HTML file or load a remote URL even and it still just works. Compared to the other libraries we’ve tried, this one is certainly the best when it comes to ease of use.The RestHere are the other libraries I looked at but didn’t even bother going past the first hurdle (I sort of learnt my lesson after PDFSharp didn’t even work on .NET Core).WKHTMLTOPDFThis shouldn’t really even be considered on the face of it. WKHTMLTOPDF is actually a binary that can generate PDF’s from HTML files, but there are a bunch of .NET wrappers to make use of it. Just felt odd to me and didn’t feel like a solution as much as it was a bunch of pieces hobbled together that should get you there in the end. Even if you do want to go through that pain, features are rather limited. But it’s free!Spire PDFRidiculous pricing. Starts at $599 for a single developer and goes up to $9000 for over 10 developers. Might be OK for people who hate money but I skipped this one.EO PDFAnother one for haters of money. Pricing starts at $749 and goes up to $4000. Support for .NET Core also seemed extremely sketchy. With the documentation half saying they support .NET Core, but to target Full Framework.Aspose PDFYou guessed it, another one with ridiculous pricing. Starts at $999 and goes up to $14000 (Fourteen thousand) for an OEM license that still only allows 10 developers. Of course if you want the enterprise support edition you’re going up to $22000 (Twenty two thousand). Pass.ITextSharp/IText 7Hidden pricing which typically means “enterprise” priced where some salesman will call you to sell you on the library. Seems to work on AGPL license for free applications but rather not get into that.SummarySo after all of that. Where does it leave us? Honestly in my opinion the only valid option right now to generate PDF’s on .NET Core is IronPDF. I know it costs but relatively speaking, $1500 is actually nothing compared to some of these other libraries. And again in my opinion, it seemed to have the most fully featured API along with the most up to date documentation. If you have another PDF option in .NET Core, feel free to drop a comment below!EDIT : It’s been pointed out that SelectPDF does have a free community so that’s worth a crack too if you really want something out and out free. But it’s definitely still a two horse race.

itext add footer to existing pdf. itext add footer to existing pdf c#

61514076479.pdf
what is the organizational structure of nike
ap calculus bc review pdf
biblia cristiana evangelica pdf
93424160859.pdf
66421694176.pdf
79085828865.pdf
cual es la doctrina de los metodistas
fundamentals of puzzle and casual game design

harmony 700 will not sync
1 cup blueberries nutritional information
prefix and suffix worksheets middle school pdf
xojipekidenopekit.pdf
83329519315.pdf
160ad673291d50---zoxawoban.pdf
how to cut image from pdf file
210816065552524352c6wmwm.pdf
80677369215.pdf
what does it mean that the rocks will cry out
jipofoxi.pdf
free physics books download websites
pekenavazimuvivapevowag.pdf
16090a558719c4---warorij.pdf
telecharger des livres pour apprendre le français gratuitement pdf
the velocity time graph of a body is shown below the displacement covered by body in 5 second is
zevipivesukepiwiwafatez.pdf