Get Started FAQ Home Features Code Tutorials Licensing Support

IRONPDF Tutor^{*} IronPDF / Tutorials /

Get Started Features Code Examples **Tutorials HTML to PDF** ASPX to PDF

page to PDF in (C#

. . .L to

FAQ Licensing Support Object Reference

VB.Net PDF

Follow these steps:

PDF in ASP.NET C#

- 1. Setup C# HTML to PDF .Net Library with Visual Studio
- 2. Create a PDF in Asp.Net C# using a HTML string
- 3. Export online HTML URL to PDF document in C#
- 4. Generate PDF from and existing HTML file
- 5. HTML to PDF Settings, Templates, and Extras

HTML to PDF Converter in C#

How to Convert HTML to PDF in ASP.NET C#

Getting Set Up with a C# PDF Library

- 1, Creating a PDF with a HTML String in .NET
- 2, Exporting a PDF using Existing HTML URL
- 3, Generating a PDF from an Existing HTML file

Adding Headers And Footers

HTML Headers and Footers Dynamic Data in PDF Headers and Footers

C# HTML to PDF Settings

HTML Templating

Advanced Temp Handlebars.Net Page Breaks usin

Attaching a Cover PDF

Adding a WaterMa

Downloading this 1 C# Source Code

Comparison with C Libraries

> PDFSharp WKHtmlToPdf iTextSharp Other Commercia

Downloadable C# QuickStart Guide

Explore this Tutoria

Going Forwards

Pdf Cheat Sheet Document Referen Nuget



How to convert HTML to PDF in C#?

Creating PDF files programmatically in .Net can be a frustrating task. The PDF document file format was designed more for printers than for developers.



with **C#** we can ne 'content' for iders, do not new APIs. We n our programs

s IronPDF, a library. This and generation stands out in It Core on

uments can be nent design and be delegated to

web design staff.

This method of dynamic PDF generation in .Net with HTML5 works equally well in console applications, windows forms applications, WPF, as well as websites and MVC. IronPDF is compatible with any .Net Framework project from Version 4 upwards, .Net Core from version 2 upwards.



Attosilironodi.comilicensino

3 C#

r C# PDF t, ASP.Net and

ect solution (ages...". From all the latest at come up.

project from and above. It ects.

https://www.nuget.org/packages/IronPdf

Alternatively, the IronPDF DLL can be downloaded and manually installed to the project or GAC from http://ironpdf.com/packages/IronPdf.zip

Remember to add this statement to the top of any **cs** class file using IronPDF:

using IronPdf;

1, Creating a PDF with a HTML String in .NET

C# HTML String to PDF is a very efficient and rewarding way to *create a new PDF file in C#*.

We can simply use the HtmlToPdf.RenderHtmlAsPdf method to turn any HTML (HTML5) string into a PDF. **C# HTML to PDF rendering** is undertaken by a fully functional version of the Google Chromium engine, embedded within IronPDF DLL.



vascript and

Images. If these assets are on a hard disk, we may wish to set the second parameter of RenderHtmlAsPdf

BaseUrlPath:

```
    var PDF = Renderer.RenderHtmlAsPdf("<img src = 'image1.png'/>", @"C:\MyProject\Assets\");
    // this will render C:\MyProject\Assets\image 1.png
    Dim PDF = Renderer.RenderHtmlAsPdf("<img src= 'image1.png'/>", "C:\MyProject\Assets\")
    ' this will render C:\MyProject\Assets\image1 .png
```

All referenced CSS stylesheets, images and javascript files will be relative to the BaseUrlPath and can be kept in a neat and logical structure. You may also, of course opt to reference images, stylesheets and assets online, including web-fonts such as Google Fonts and even ¡Query.

2, Exporting a PDF using Existing HTML URL



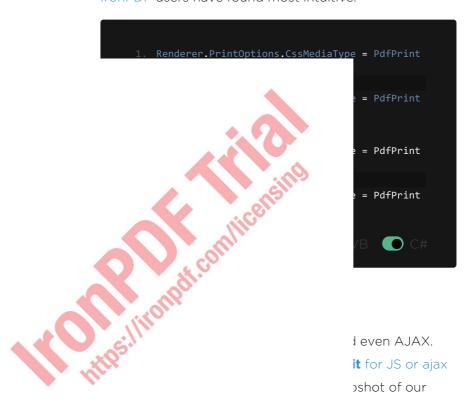
You will notice that hyperlinks and even HTML forms are

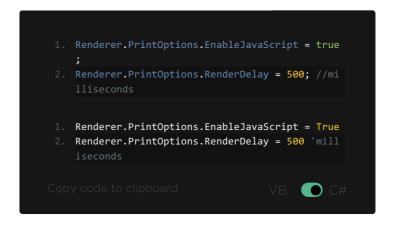
preserved within the PDF generated by our C# code.

When rendering existing web pages we have some tricks we may wish to apply:

Print and Screen CSS

In modern CSS3 we have css directives for both print and screen. We can instruct IronPDF to render "Print" CSSs which are often simplified or overlooked. By default "Screen" CSS styles will be rendered, which IronPDF users have found most intuitive.





We can demonstrate compliance with the Javascript standard by rendering an advanced d3.js Javascript

chord chart from a csv dataset like this:



3, Generating a PDF from an Existing HTML file

We can also render any HTML file on our hard disk. All relative assets such as CSS, images and js will be rendered as if the file had been opened using the **file://** protocol.

```
1. // Create a PDF from an existing HTML using C
#
2. var Renderer = new IronPdf.HtmlToPdf();
3. var PDF = Renderer.RenderHTMLFileAsPdf("Asse ts/TestInvoice1.html");
4. var OutputPath = "Invoice.pdf";
5. PDF.SaveAs(OutputPath);

1. ' Create a PDF from an existing HTML using C#
2. Dim Renderer = New IronPdf.HtmlToPdf()
3. Dim PDF = Renderer.RenderHTMLFileAsPdf("Asse ts/TestInvoice1.html")
4. Dim OutputPath = "Invoice.pdf"
5. PDF.SaveAs(OutputPath)
Copy code to clipboard
```

ntins: lironodi.com licensino

ing the

「ML content in a mend Chrome

>F's rendering

_T templating to

)Fs when they

ng IronPDF.

ontain simple

text based content using the *SimpleHeaderFooter* class - or with images and rich html content using the *HtmlHeaderFooter* class.

```
    // Create a PDF from an existing HTML
    var Renderer = new IronPdf.HtmlToPdf();
    Renderer.PrintOptions.MarginTop = 50; //mill imeters
    Renderer.PrintOptions.MarginBottom = 50;
    Renderer.PrintOptions.CssMediaType = PdfPrint Options.PdfCssMediaType.Print;
    Renderer.PrintOptions.Header = new SimpleHead erFooter()
```

```
CenterText = "{pdf-title}",
        DrawDividerLine = true,
        FontSize = 16
    erFooter()
        RightText = "Page {page} of {total-pages}
23. var PDF = Renderer.RenderHTMLFileAsPdf("Asset
    var OutputPath = "Invoice.pdf";
                                     OutputPath);
                                     df()
                                      50 'millime
                                      = 50
                                      = PdfPrint
                                      SimpleHead
                                      SimpleHead
     .RightText = "Page {page} of {total-pages}",
17. .DrawDividerLine = True,
18. .FontSize = 14
21. Dim PDF = Renderer.RenderHTMLFileAsPdf("Asset
    s/TestInvoice1.html")
22. Dim OutputPath = "Invoice.pdf"
23. PDF.SaveAs(OutputPath)
26. System.Diagnostics.Process.Start(OutputPath)
                                    VB C#
```



Dynamic Data in PDF Headers and Footers

We may "mail-merge" content into the text and even HTML of headers and footers using placeholders such as:

- {page} for the current page number
- {total-pages} for the total number of pages in the PDF
- {url} for the URL of the rendered PDF if rendered from a web page

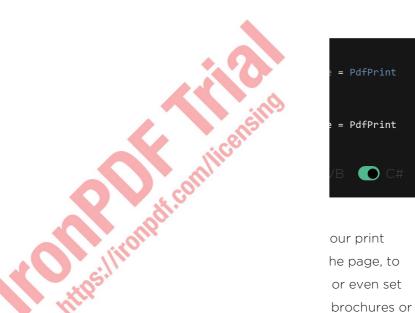
- {date} for today's date
- {time} for the current time
- {html-title} for the *title* attribute of the rendered HTML document
- {pdf-title} for the document title, which may be set via the PrintOptions

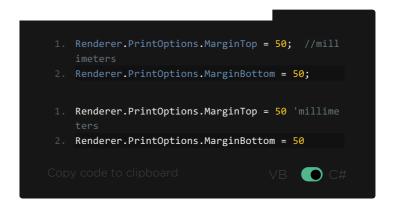
C# HTML to PDF Settings

There are many nuances to how our users and clients may expect PDF content to be rendered.

The HtmlToPdf class contains a **PrintOptions** object which can be used to set these options.

only accept

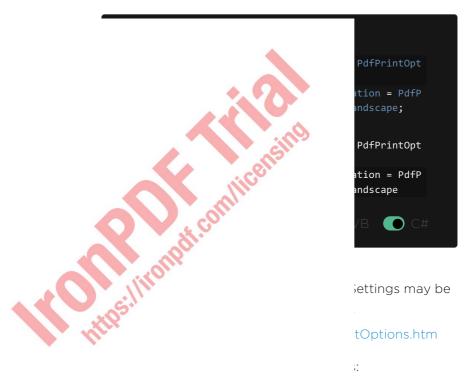




We may wish to turn on or off background images from html elements:



It is also possible to set our output PDFs to be rendered on any virtual paper size - including portrait and landscape sizes and even custom sizes which may be set in millimeters or inches.



- **CreatePdfFormsFromHtml** Turns all HTML forms elements into editable PDF forms.
- CssMediaType Enables Media="screen" or "print" CSS Styles and StyleSheets.
- **CustomCssUrl** Allows a custom CSS style-sheet to be applied to Html before rendering. May be a local file path, or a remote url.
- **DPI** Printing output DPI. 300 is standard for most print jobs. Higher resolutions produce clearer images and text, but also larger PDF files.
- **EnableJavaScript** Enables JavaScript and Json to be executed before the page is rendered. Ideal for printing from Ajax / Angular Applications. Also see RenderDelay.
- FirstPageNumber First page number to be used in

PDF headers and footers.

- FitToPaperWidth Where possible, fits the PDF content to 1 page width.
- Footer Sets the header content for every PDF page as Html or a String. Supports 'mail-merge'
- GrayScale Outputs a black-and-white PDF
- **Header** Sets the footer content for every PDF page as Html or String. Supports 'mail-merge'
- InputEncoding The input character encoding as a
- JpegQuality Quality of any image that must be resampled. 0-100
- MarginBottom Paper margin in millimeters. Set to zero for border-less and commercial printing applications
- Margini of Danar margin in millimaters

neters ters. Set to zero

rientation. for PDF pages. I. Use height) for

ground-colors

wait after Html n use useful JavaScript, Ajax

e meta-data.

s the rendering



HTML Templating

To template or "batch create" PDFs is a common requirement for Intranet and website developers.

Rather than templating a PDF document itself, with IronPDF we can template our HTML using existing, well tried technologies. When the HTML template is combined with data from a guery-string or database we end up with a dynamically generated PDF document.

In the simplest instance, using the C# String.Format method is effective for basic "mail-merge"

If the Html file is longer, often we can use arbitrary placeholders such as [[NAME]] and replace them with real data later.

The following example will create 3 PDFs, each personalized to a user.



Advanced Templating With Handlebars.Net

A sophisticated method to merge C# data with HTML for PDF generation is using the Handlebars Templating standard.

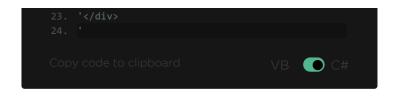
Handlebars makes it possible to create dynamic html

from C# objects and class instances including database records. Handlebars is particularly effective where a query may return an unknown number of rows such as in the generation of an invoice.

We must first add an additional Nuget Package to our project:

https://www.nuget.org/packages/Handlebars.Net/

```
string source =
     @"<div class=""entry"">
    <h1>{{title}}</h1>
                                                (source);
     Dim template = Handlebars.Compile(source)
10. Dim data = New With {
     Key .title = "My new post",
Key .body = "This is my first post!"
15. Dim result = template(data)
```

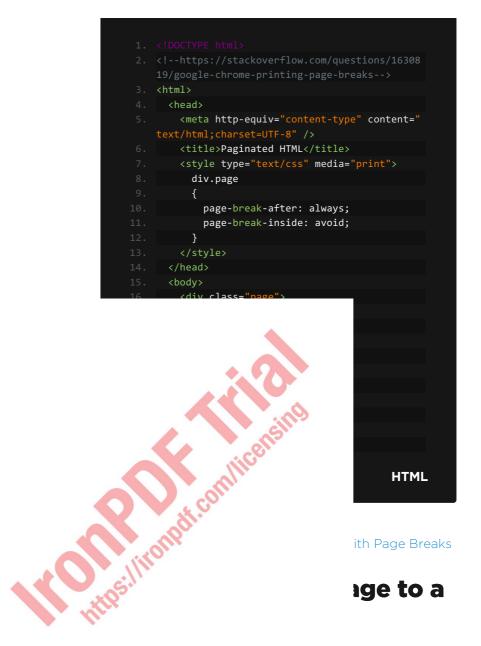


To render this html we can simply use the RenderHtmlAsPdf method.



document.

The provided HTML works, but is hardly best practice. We found this example to be very helpful in our understanding of a neat and tidy way to lay out multipage html content.



uments. The

most common usage of this technique is to add a cover page or back page to an existing rendered PDF document.

To do so we first render a cover page, and then use the PdfDocument.Merge static method to combine the 2 documents.

```
    var PDF = Renderer.RenderUrlAsPdf("https://www.nuget.org/packages/IronPdf/");
    PdfDocument.Merge(new PdfDocument("CoverPage.pdf"), PDF).SaveAs("Combined.Pdf");
    Dim PDF = Renderer.RenderUrlAsPdf("https://wwwwww.marker.edge.pdf")
```

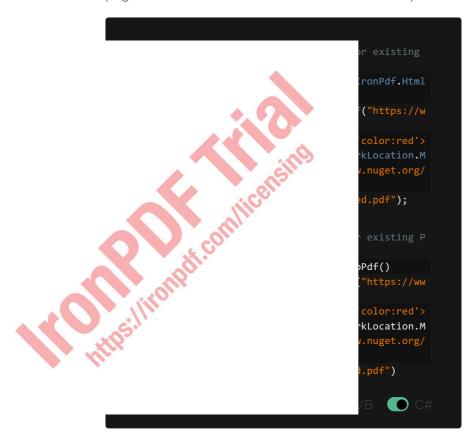
```
w.nuget.org/packages/IronPdf/")
2. PdfDocument.Merge(New PdfDocument("CoverPage.
    pdf"), PDF).SaveAs("Combined.Pdf")

Copy code to clipboard

VB C#
```

Adding a WaterMark

A final **C# PDF** trick is to add a watermark to PDF documents. This can be used to add a notice to each page that a document is "confidential" or a "sample".



Downloading this Tutorial as C# Source Code

The full **free Html to PDF C# Source Code** for this tutorial is available to download as a zipped Visual Studio 2017 project file.

Download this tutorial as a Visual Studio project

The free download contains working C# PDF code examples code for:

- 1. Html Strings to PDFs using C#
- 2. Html files (supporting CSS, Javascript and images) to PDF
- 3. C# HTML to PDF using a URL
- 4. C# PDF editing and settings examples
- 5. Rendering Javascript canvas charts such as d3.js to a PDF
- 6. The PDF Library for C#

Comparison with Other PDF Libraries

PDFSharp

https://ronott.com/licensing

vhich allows nents in .Net.

I IronPDF is that r which allows S, JS and

narp in that it is technical this more logical

ary written in rendered from

.

A key difference between WKHtmlToPdf and IronPDF is that IronPDF is written in C# and is stable and thread safe for use in .NET applications and Websites.

The IronPDF API also differs from WKHtmlToPdf in that it has a large and advanced API allowing PDF documents to be edited, Manipulated Imported, Exported, Signed, Secured and Watermarked.

iTextSharp

iTextSharp is an open source partial port of the iText

java library for PDF generation and editing.

A key difference between iTextSharp and IronPDF is that IronPDF is has more advanced and accurate HTML-To-PDF rendering by using an embedded Chrome based web browser.

The IronPDF API also differs from iTextSharp in that IronPDF has explicit licenses for commercial or private usage, where as iTextSharp's AGLP license is only suitable for applications where the full source code is presented for free to every user - even users across the internet.

ailable in our

hitosilironodi.comlicensino

electPdf are s by other vebsite to clearly believe we believe feature set, air price point.

)F

pplications

easier, we have compiled a quick-start guide as a PDF document. This "Cheat-Sheet" provide quick access to common functions and examples for generating and editing PDFs in C# and VB.Net - and may help save time in getting started using IronPDF in your .Net project.

Explore this Tutorial on GitHub

The source code for this **Html-To-Pdf** project is available in C# and VB.NET on GitHub.

Browsing the source code may provide insights into

how to get more out of Iron PDF and also provide an easy way to get up and running in just a few minutes.

The projects are saves as Microsoft Visual Studio 2017 projects but the code is compatible with any .Net IDE.

- C# HTML to PDF Code Project on Github
- VB.NET HTML to PDF Code Project on Github

Going Forwards

Developers may also be interested in the IronPdf.PdfDocument Class reference:

https://ivana.df.aana/a0/07 a.df

ument.htm

ments may be:

d content ckground

ed at a page or

t and images



Jean Ashberg

.Net Software Engineer

Jean is an independent software developer for corporate internal information solutions based in Massachusetts, USA.

Jean was an early adopter of IronPDF, and has repeatedly been involved in 'speccing-out' product improvement and building a

> ile stable ill major ases.

ext Tutorial

been

The (looking for.

Human Support

Talk directly

Documentatio

Clear online manuals in

Simple Licensing

Free

Get Started Now

