IRONPDF

Tutor

HTML to PDF Converter in C#

page to PDF in
(C#

L to PDF in ASP.NET C#

Follow these steps:

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

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
oders, do not
new APIs. We
n our programs

s IronPDF, a
library. This
ınd generation
stands out in
ɪt 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.

a C#

r C# PDF
t, ASP.Net and

ect solution
kages...". 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.

```
// Render any HTML fragment or document to HT
ML
var Renderer = new IronPdf.HtmlToPdf();
var PDF = Renderer.RenderHtmlAsPdf("<h1>Hello
IronPdf</h1>");
```

```
file so we c
PDF viewer
OutputPath);

ument to HTM

Pdf()
f("<h1>Hello

ile so we ca
F viewer
OutputPath)
```

B C#

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
```

Copy code to clipboard VB C#

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 jQuery.

2, Exporting a PDF using Existing HTML URL

:# is very
ams to split
g work across

in the following

```
web page
Pdf();
("https://en
ment_Format"

file so we c

'wikipedia.p
```

```vb
' Create a PDF from any existing web page
Dim Renderer = New IronPdf.HtmlToPdf()
Dim PDF = Renderer.RenderUrlAsPdf("https://en.wikipedia.org/wiki/Portable_Document_Format")
PDF.SaveAs("wikipedia.pdf")

' This neat trick opens our PDF file so we can see the result
System.Diagnostics.Process.Start("wikipedia.pdf")
```

Copy code to clipboard VB C#

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.

```
Renderer.PrintOptions.CssMediaType = PdfPrint
e = PdfPrint
```

```
e = PdfPrint
e = PdfPrint
```

VB C#

d even AJAX.
it for JS or ajax
pshot of our

```
Renderer.PrintOptions.EnableJavaScript = true
;
Renderer.PrintOptions.RenderDelay = 500; //mi
lliseconds
```

```
Renderer.PrintOptions.EnableJavaScript = True
Renderer.PrintOptions.RenderDelay = 500 'mill
iseconds
```

Copy code to clipboard VB C#

We can demonstrate compliance with the Javascript standard by rendering an advanced d3.js Javascript chord chart from a csv dataset like this:

```
// Create a PDF Chart a live rendered dataset using d3.js and javascript
var Renderer = new HtmlToPdf();
var PDF = Renderer.RenderUrlAsPdf("https://bl.ocks.org/mbostock/4062006");
PDF.SaveAs("chart.pdf");
```

```
' Create a PDF Chart a live rendered dataset using d3.js and javascript
Dim Renderer = New HtmlToPdf()
Dim PDF = Renderer.RenderUrlAsPdf("https://bl.ocks.org/mbostock/4062006")
PDF.SaveAs("chart.pdf")
```

/B C#

›e viewed in a
›rowser window
responsive
.

/pes to navigate
be responsive.

```
e = PdfPrint
```

```
e = PdfPrint
```

Copy code to clipboard VB C#

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.

```csharp
// Create a PDF from an existing HTML using C#
var Renderer = new IronPdf.HtmlToPdf();
var PDF = Renderer.RenderHTMLFileAsPdf("Assets/TestInvoice1.html");
var OutputPath = "Invoice.pdf";
PDF.SaveAs(OutputPath);
```

```vb
' Create a PDF from an existing HTML using C#
Dim Renderer = New IronPdf.HtmlToPdf()
Dim PDF = Renderer.RenderHTMLFileAsPdf("Assets/TestInvoice1.html")
Dim OutputPath = "Invoice.pdf"
PDF.SaveAs(OutputPath)
```

Copy code to clipboard VB C#

ing the
TML content in a
ımend 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.

```csharp
// Create a PDF from an existing HTML
var Renderer = new IronPdf.HtmlToPdf();

Renderer.PrintOptions.MarginTop = 50;  //millimeters
Renderer.PrintOptions.MarginBottom = 50;
Renderer.PrintOptions.CssMediaType = PdfPrintOptions.PdfCssMediaType.Print;

Renderer.PrintOptions.Header = new SimpleHeaderFooter()
```

```
{
    CenterText = "{pdf-title}",
    DrawDividerLine = true,
    FontSize = 16
};

Renderer.PrintOptions.Footer = new SimpleHead
erFooter()
{
    LeftText = "{date} {time}",
    RightText = "Page {page} of {total-pages}
",
    DrawDividerLine = true,
    FontSize = 14
};

var PDF = Renderer.RenderHTMLFileAsPdf("Asset
s/TestInvoice1.html");
var OutputPath = "Invoice.pdf";

                                  file so we c

                                  OutputPath);

                                  TML
                                  df()

                                   50 'millime

                                  n = 50
                                  e = PdfPrint

                                  SimpleHead

                                  SimpleHead

 .RightText = "Page {page} of {total-pages}",
 .DrawDividerLine = True,
 .FontSize = 14
}

Dim PDF = Renderer.RenderHTMLFileAsPdf("Asset
s/TestInvoice1.html")
Dim OutputPath = "Invoice.pdf"
PDF.SaveAs(OutputPath)

' This neat trick opens our PDF file so we ca
n see the result
System.Diagnostics.Process.Start(OutputPath)
```

Copy code to clipboard VB C#

```csharp
Renderer.PrintOptions.Footer = new SimpleHeaderFooter()
{
    LeftText = "{date} {time}",
    RightText = "Page {page} of {total-pages}",
    DrawDividerLine = true,
    FontSize = 14
};
```

```vb
Renderer.PrintOptions.Footer = New SimpleHeaderFooter() With {
 .LeftText = "{date} {time}",
 .RightText = "Page {page} of {total-pages}",
 .DrawDividerLine = True,
 .FontSize = 14
}
```

/B C#

ich headers and
ntent which
d hyperlinks.

```
v HtmlHeader
tyle='text-a
'>page {page
```

```
v HtmlHeader
div style='t
pink'>page
>"}
```

Copy code to clipboard 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

```
= PdfPrint
```

```
= PdfPrint
```

VB C#

our print
he page, to
or even set
brochures or

```csharp
Renderer.PrintOptions.MarginTop = 50;  //millimeters
Renderer.PrintOptions.MarginBottom = 50;
```

```vb
Renderer.PrintOptions.MarginTop = 50 'millimeters
Renderer.PrintOptions.MarginBottom = 50
```

Copy code to clipboard VB C#

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

```
Renderer.PrintOptions.PrintHtmlBackgrounds =
true;
```

```
Renderer.PrintOptions.PrintHtmlBackgrounds =
True
```

Copy code to clipboard VB C#

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.

```
PdfPrintOpt
ation = PdfP
andscape;
```

```
PdfPrintOpt
ation = PdfP
andscape
```

VB C#

ettings may be

tOptions.htm

:

- **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 string
- **JpegQuality** Quality of any image that must be re-sampled. 0-100
- **MarginBottom** Paper margin in millimeters. Set to zero for border-less and commercial printing applications
- **MarginLeft** Paper margin in millimeters

neters
ters. Set to zero
nting

rientation.
e for PDF pages.
d. 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 query-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"

```
String.Format("<h1>Hello {0} !<h1>","World");
```

```
String.Format("<h1>Hello {0} !<h1>","World")
```

Copy code to clipboard VB C#

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.

```
>";
", "Jenny"
e.Replace("
sPdf(HtmlIn
>"
nny" }
Replace("[[N
f(HtmlInsta
 Pdf.SaveAs(name & ".pdf")
Next name
```

Copy code to clipboard VB C#

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>
  <div class=""body"">
    {{body}}

                                     (source);

                                     "

                                     =""entry"">

  </div>
</div>"

Dim template = Handlebars.Compile(source)

Dim data = New With {
 Key .title = "My new post",
 Key .body = "This is my first post!"
}

Dim result = template(data)

' Would render:
'<div class="entry">
'  <h1>My New Post</h1>
'  <div class="body">
'    This is my first post!
'  </div>
```

```
'</div>
'
```

Copy code to clipboard VB C#

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

```
IronPdf.HtmlToPdf Renderer = new IronPdf.Html
ToPdf();
Renderer.RenderHtmlAsPdf(HtmlInstance).SaveAs
("Handelbars.pdf")
```

```
pPdf()
ance).SaveAs
```

/B C#

rs html
om
et

nent is for
where PDF
e layout.

known CSS trick
printed HTML
document.

```html
<div style='page-break-after: always;'>&nbsp;
</div>
```

Copy code to clipboard HTML

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 multi-page html content.

```html
<!DOCTYPE html>
<!--https://stackoverflow.com/questions/1630819/google-chrome-printing-page-breaks-->
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;charset=UTF-8" />
    <title>Paginated HTML</title>
    <style type="text/css" media="print">
      div.page
      {
        page-break-after: always;
        page-break-inside: avoid;
      }
    </style>
  </head>
  <body>
    <div class="page">
```

HTML

ith Page Breaks

age to a

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.

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

```vb
Dim PDF = Renderer.RenderUrlAsPdf("https://ww
w.nuget.org/packages/IronPdf/")
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".

```
r existing
IronPdf.Html
("https://w
'color:red'>
kLocation.M
.nuget.org/
d.pdf");
 existing P
Pdf()
("https://ww
'color:red'>
kLocation.M
.nuget.org/
.pdf")
```

VB C#

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

vhich allows
nents in .Net.

d IronPDF is that
r which allows
S, JS and

harp in that it is
technical
this more logical

rary written in
rendered from
HTML.

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

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://ironpdf.com/c%23-pdf

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

gle stable

ll major

ases.

ext Tutorial

The C been looking for.

Human Support

Talk directly

Documentatio

Clear online manuals in

Simple Licensing

Free

Get Started Now

with our development team

Ask a Question

plain English.

View Documentati

development license. Commercial from $399.

Browse Options

Get started in minutes with NuGet or DLL.

Install & Tr

Information

Ask a Question

Documentation

Tutorials

Object Reference