

Tutorial

How to create PDFs in ASP .NET with ASPX to PDF Converter

ASPX to PDF

```
1. using System;  
2. using System.Collections.Generic;  
3. using System.Linq;  
4. using System.Web;  
5. using System.Web.UI;  
6. using System.Web.UI.WebControls;  
8. using IronPdf;  
7. namespace AspxToPdfTutorial
```

ASPX Pages to PDF in ASP .NET

by Jacob Müller

Interact with the tutorial: <https://ironpdf.com/tutorials/aspx-to-pdf/>

Share the tutorial: [✉](#) [f](#) [@](#) [in](#) [t](#)

This ASPX to PDF tutorial will guide you step-by-step how to save an ASPX page as a PDF in ASP.NET web applications. Apply settings including setting file behavior and names, adding headers & footers, changing print options, adding page breaks, combining Async & Multithreading, and more.

Table of Contents

1. **Install the C# Library Free from IronPDF**
2. **Convert ASP.NET Webpages to PDF**
3. **Apply ASPX File to PDF Converter Settings**
4. **Add Headers & Footers to ASPX PDFs**
5. **Apply ASPX File to PDF Tricks: Page Breaks**
6. **Combine Async & Multithreading for Performance**
7. **Download as ASP.NET Source Code**

ASP.NET to PDF Conversion

Follow these guiding steps:

1. Install IronPDF, an ASPX to PDF Converter, into Visual Studio
2. Start by converting an ASPX web form. See example Invoice.aspx
3. Configure ASPX to PDF Settings for your example
4. Add optional Headers and Footers to PDF document
5. Trigger Page Breaks and use Async + Multithreading
6. Download the ASPX page in PDF format using C#


Microsoft Web Form Applications for ASP.Net are commonly used in the development of sophisticated websites, online banking, intranets and accounting systems. One common feature of ASP.Net (ASPX) websites is to generate dynamic PDF files such as invoices, tickets, or management reports for users to download in PDF format.

This tutorial shows how to use the IronPDF software component for .NET to turn any ASP.Net web form into a PDF (ASP .Net to PDF). HTML, normally rendered as a web page, will be used to render as a PDF for download or viewing in a web browser. The attached source project will show you how to convert a webpage to PDF in Asp .net using C#.

We achieve this HTML to PDF conversion (ASPX to PDF) when rendering webpages using the IronPDF library and its [AspxToPdf](#) Class.

Step 1

1. Install the C# Library Free from IronPDF



Download DLL
Manually install into your project

or



Install with NuGet
nuget.org/packages/IronPdf/

Install via NuGet

In Visual Studio, right click on your project solution explorer and select "Manage Nuget Packages...". From there simply search for IronPDF and install the latest version... click ok to any dialog boxes that come up.

This will work in any C# .Net Framework project from Framework 4 and above, or .Net Core 2 and above. It will also work just as well in VB.Net projects.

```
PM > Install-Package IronPdf
```

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

Install via DLL

Alternatively, the IronPDF DLL can be downloaded and manually installed to the project or GAC from

<https://ironpdf.com/packages/IronPdf.zip>

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

```
using IronPdf;
```

How to Tutorials

2. Create a PDF with an HTML String in .NET C#

We start with a normal ASPX "Web Form," which renders as HTML. We later **convert the ASPX page to PDF** file format.

In the attached example source code, we rendered a business invoice "Invoice.aspx," a simple HTML business invoice rendered as an ASP .NET Page.

The HTML page contains CSS3 stylesheets and may also include images and javascript.

To render this ASP.NET Web Page to a PDF instead of HTML, we need to open the C# (or VB.Net) code and add this to the *Page_Load* event:

```
1. IronPdf.AspxToPdf.RenderThisPageAsPdf(IronPdf.AspxToPdf.FileBehavior.InBrowser);
```

This is all that's required; the HTML now renders as a PDF. Hyperlinks, StyleSheets, Images and even HTML forms are preserved. This is very similar to the output if the user printed the HTML to a PDF in their browser themselves. IronPDF is built upon Chromium web browser technology that powers Google Chrome.

The entire C# code reads like this in full:

```
1. using System;
2. using System.Collections.Generic;
3. using System.Linq;
4. using System.Web;
5. using System.Web.UI;
6. using System.Web.UI.WebControls;
7. using IronPdf;
8.
9. namespace AspxToPdfTutorial
10. {
11.     { public partial class Invoice : System.Web.UI.Page
12.         {
13.             protected void Page_Load(object sender, EventArgs e)
14.             {
15.                 IronPdf.AspxToPdf.RenderThisPageAsPdf(IronPdf.AspxToPdf.FileBehavior.InBrowser);
16.             }
17.         }
18.     }
```

3. Apply ASPX File to PDF Converter Settings

There are many options to tweak and perfect when we convert an ASPX file to PDF generated using .Net Web Forms.

These options are documented in full online at

https://ironpdf.com/c%23-pdf-documentation/html/M_IronPdf_AspxToPdf_RenderThisPageAsPdf.htm

3.1. Set PDF File Behavior

"InBrowser" file behavior attempts to show the PDF directly in the user's browser. This is not always possible in every web browser, but typically a common feature of modern, standards-compliant browsers.

```
IronPdf.AspxToPdf.RenderThisPageAsPdf(IronPdf.AspxToPdf.FileBehavior.InBrowser);
```

"Attachment" file behavior causes the PDF to be downloaded.

```
IronPdf.AspxToPdf.RenderThisPageAsPdf(IronPdf.AspxToPdf.FileBehavior.Attachment);
```

3.2. Set PDF File Name

We may also set the file name of the PDF document by adding an additional parameter. This means we can control the name of the file when the user decides to download or keep it. When we save the ASPX page as a PDF, this name will be given to the PDF document.

```
IronPdf.AspxToPdf.RenderThisPageAsPdf(IronPdf.AspxToPdf.FileBehavior.Attachment, "Invoice.pdf");
```

3.3. Change PDF Print Options

We can control the output of the PDF by adding an instance of the `IronPdf.PdfPrintOptions` Class:

https://ironpdf.com/c%23-pdf-documentation/html/T_IronPdf_PdfPrintOptions.htm

```
1. var AspxToPdfOptions = new IronPdf.PdfPrintOptions()
2.     {
3.         DPI = 300,
4.         EnableJavaScript = false,
5.         //.. many more options available
6.     };
7. IronPdf.AspxToPdf.RenderThisPageAsPdf(IronPdf.AspxToPdf.FileBehavior.Attachment, "Invoice.pdf",
8. AspxToPdfOptions);
```

The PDF PrintOptions available include:

- **CreatePdfFormsFromHtml** Turns ASPX form elements into editable PDF forms
- **CssMediaType** Enables Media="screen" or "print" for CSS Styles and CSS3 StyleSheets
- **CustomCssUrl** Allows a custom CSS style-sheet to be applied to HTML by URL
- **DPI** Output DPI resolution of the PDF
- **EnableJavaScript** Enables JavaScript, jQuery and even Json code within the ASPX Page. A `RenderDelay` may need to be applied
- **FirstPageNumber** First page number for Header and Footer. The default is 1
- **FitToPaperWidth** Where possible, shrinks the PDF content to a width of 1 page of virtual paper
- **Footer** Sets the footer content for every PDF page using content strings or even HTML
- **GrayScale** Outputs a greyscale PDF in shades of grey instead of full color
- **Header** Sets the header content for every PDF page using content strings or even HTML
- **InputEncoding** The input character encoding as a string. UTF-8 is Default for ASP.NET
- **JpegQuality** Quality of any image within the ASPX Page which is large enough to need to be re-sampled. Values range from 0-100. 100 is highest quality, but also largest file size
- **MarginBottom** Bottom PDF Paper margin in millimeters. Set to zero for a borderless pdf
- **MarginLeft** Left PDF Paper margin in millimeters. Set to zero for a borderless pdf
- **MarginRight** Right PDF Paper margin in millimeters. Set to zero for a borderless pdf
- **MarginTop** Top PDF Paper margin in millimeters. Set to zero for a borderless pdf
- **PaperOrientation** The PDF paper orientation. Landscape or Portrait
- **PaperSize** Set an output paper size for PDF pages using `System.Drawing.Printing.PaperKind`. Alternatively developers may use the `SetCustomPaperSize(int width, int height)` method to create custom sizes
- **PrintHtmlBackgrounds** Prints HTML image backgrounds
- **RenderDelay** Milliseconds to wait after Html is rendered before printing so that Javascript or JSON have time to work
- **Title** PDF Document 'Title' meta-data
- **Zoom** A % Scale level allowing the developer to enlarge or shrink html content

4. Add Headers & Footers to ASPX PDFs

Using IronPDF, Headers and Footers can be added to the PDF output.

The simplest way to do this is with the SimpleHeaderFooter class, which supports a basic layout that can easily add dynamic data such as the current time and page numbering.

4.1. ASPX to PDF Header & Footer Example

```
1. using System;
2. using System.Collections.Generic;
3. using System.Linq;
4. using System.Web;
5. using System.Web.UI;
6. using System.Web.UI.WebControls;
7. namespace AspxToPdfTutorial
8. {
9.     public partial class Invoice : System.Web.UI.Page
10.    {
11.        protected void Page_Load(object sender, EventArgs e)
12.        {
13.            var AspxToPdfOptions = new IronPdf.PdfPrintOptions()
14.            {
15.                Header = new IronPdf.SimpleHeaderFooter()
16.                {
17.                    CenterText = "Invoice",
18.                    DrawDividerLine = false,
19.                    FontFamily = "Arial",
20.                    FontSize = 12
21.                },
22.                Footer = new IronPdf.SimpleHeaderFooter()
23.                {
24.                    LeftText = "{date} - {time}",
25.                    RightText = "Page {page} of {total-pages}",
26.                    FontFamily = "Arial",
27.                    FontSize = 12,
28.                },
29.            };
30.
31.
32. IronPdf.AspxToPdf.RenderThisPageAsPdf(IronPdf.AspxToPdf.FileBehavior.Attachment, "Invoice.pdf",
33. AspxToPdfOptions);
34.     }
35. }
36. }
```

Alternatively we can generate HTML headers and footers using the `HtmlHeaderFooter` class, which also supports CSS, images, and hyperlinks.

```
1. using System.Collections.Generic;
2. using System.Linq;
3. using System.Web;
4. using System.Web.UI;
5. using System.Web.UI.WebControls;
6. namespace AspxToPdfTutorial
7. {
8.     public partial class Invoice : System.Web.UI.Page
9.     {
10.        protected void Page_Load(object sender, EventArgs e)
11.        {
12.            var AspxToPdfOptions = new IronPdf.PdfPrintOptions()
13.            {
14.                Header = new IronPdf.SimpleHeaderFooter()
15.                {
16.                    MarginTop = 50, // make sufficient space for an HTML header
17.                    Header = new IronPdf.HtmlHeaderFooter()
18.                    {
19.                        HtmlFragment = "<div style='text-align:right'><em style='color:pink'>page
20. {page} of {total-pages}</em></div>"
21.                    }
22.                };
23.                IronPdf.AspxToPdf.RenderThisPageAsPdf(IronPdf.AspxToPdf.FileBehavior.Attachment,
24. "MyDocument.pdf", AspxToPdfOptions);
25.            }
26.        }
27.    }
```

As seen in our examples, we may "merge" dynamic text or html into Headers / Footers using placeholders:

- {page} for the current page number of the PDF
- {total-pages} as the total number of pages within the PDF
- {date} for today's date in a format appropriate to the server's system environment
- {time} for the time in hours:seconds using a 24 hour clock
- {html-title} inserts the title from the head tag of the ASPX web form
- {pdf-title} for the document file name

5. Apply ASPX File to PDF Tricks: Page Breaks

Where as HTML commonly 'flows' into a long page, PDFs simulate digital paper and are broken into consistent pages. Adding the following code to your ASPX page will automatically create a page-break in the .NET generated PDF.

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


6. Combine Async & Multithreading for Performance

IronPDF was built for .NET Framework 4.0, or .NET Core 2 or above. In Framework 4.5 or above projects, [ASYNC](#) can be utilised to improve performance when working with multiple documents.

Combining Async with multithreaded CPUs and the `Parallel.ForEach` command will improve bulk processing significantly.

7. Download as ASP.NET Source Code

The full **ASPX File to PDF Converter Source Code** for this tutorial is available to be downloaded as a zipped Visual Studio Web Application project.

[Download this tutorial as a ASP.Net Visual Studio project](#)

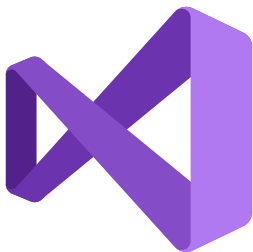
The free download contains working code examples for a C# ASP.Net Web Forms project showing a web page rendered as a PDF with settings applied. We hope this tutorial has helped you to learn how to save an [ASPX page as PDF](#).

Going Forwards

Generally, the best way to learn any programming technique is through experimentation within your own ASP.NET projects. This includes trying the ASPX to PDF Converter from IronPDF.

Developers may also be interested in the `IronPdf.AspxToPdf` Class reference:
https://ironpdf.com/c%23-pdf-documentation/html/T_IronPdf_AspXToPdf.htm

Tutorial Quick Access



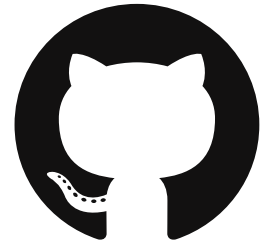
Download this Tutorial as Source Code

The full ASPX File to PDF Converter Source Code for this tutorial is available as a zipped Visual Studio Web Application project. The free download contains working code examples for a C# ASP.Net Web Forms project, showing a web page rendered as a PDF with settings applied.

[Download](#)

Explore this Tutorial on GitHub

The code for this C# ASPX-To-PDF project is available in C# and VB.NET on GitHub as a ASP.Net website project. Please go ahead and fork us on Github for more help using IronPDF. Feel free to share this with anyone who might be asking, 'How do I Convert ASPX to PDF?'

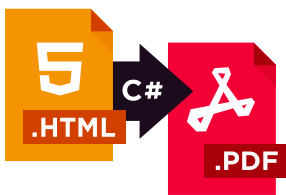


[C# ASPX to PDF Website Project >](#)

[Advanced ASP .Net Page to PDF Samples in C# for creating PDFs >](#)

[ASP.Net PDF Examples in VB.NET for creating PDFs >](#)

Download C# PDF Quickstart guide



To make developing PDFs in your .NET applications 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 will help save time getting started using IronPDF in your .NET project.

[Download](#)

View the object reference

Explore the Object Reference for IronPDF, outlining the details of all of IronPDF's features, namespaces, classes, methods fields and enums.

[View the Object Reference >](#)



The C# PDF solution you've been looking for.



Support

Open a support ticket with our development team.

[Ask a Question](#)



Documentation

View code examples and tutorials.

[Get Started](#)



Licensing

Free for development. License from \$399.

[See Licenses](#)



Try IronPDF Free

Get set up in 5 minutes.

[Download](#)