IRONPDF

Tutorial

How to Get Started with the IronPDF C# PDF Library

HTML String to PDF

```
using IronPdf;
namespace IronPDFSample
{
    class Program
    {
        static void Main()
        {
            var Renderer = new HtmlToPdf();
```

IRONPDF

IronPDF C# PDF Library

Interact with the tutorial: https://ironpdf.com/docs/

Share the tutorial:

IronPDF takes care of the difficult problem of adding PDF generation to your app, and automates turning formatted documents into a PDF.

- Convert web forms, local HTML pages, and other web pages to PDF with .NET
- Allow users to download documents, send them by email, or store them in the cloud.
- Produce invoices, quotes, reports, contracts, and other documents.
- Work with ASP.NET, ASP .NET Core, web forms, MVC, Web APIs on .NET Framework, and .NET Core.

Table of Contents

1. Install the IronPDF C# Library to your project

Download DLL

Manually install into your project

or

Install with NuGet

nuget.org/packages/IronPdf/

1.1. Install with NuGet Package Manger

Install IronPDF in Visual Studio or at the command line with the NuGet Package Manager. In Visual Studio, navigate to the console with:

- {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

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

And check out IronPDF on NuGet for more about version updates and installation.

1.2. Directly Download the DLL

Alternatively, you can directly download the DLL.

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

```
using IronPdf;
```

1.3. Install and Deploy the Library

For more details, check the guide on how to install and deploy the IronPDF C# Library.

2. Render HTML String to PDF

IronPDF can render HTML text to PDF quite easily. This example illustrates the capability. Use this option when you only need to add simple text to your PDF document.

- Create a new .NET Core console application
- Install the NuGet package
- Import the `IronPdf` namespace with the `using` keyword
- Create a new `HtmlToPdf` renderer
- Call `RenderHtmlAsPdf` and then `SaveAs` on the result.

```
1.  using IronPdf;
2.
3.  namespace IronPDFSample
4.  {
5.      class Program
6.      {
7.          static void Main()
8.          {
9.              var Renderer = new HtmlToPdf();
10.             Renderer.RenderHtmlAsPdf("<h1>Hello World<h1>").SaveAs("html-string.pdf");
11.         }
12.     }
13. }
```

3. Convert HTML File to PDF

You can render HTML files with images, CSS, forms, hyperlinks, and JavaScript as a PDF document. Use this method for scenarios where you have access to the source document locally.

This example calls `RenderHtmlAsPdf` , which returns a variable called PDF.

Call `SaveAs` to save the output to a PDF file.

The sample assumes that there is a HTML file in the folder Assets.

```
1.  using IronPdf;
2.
3.  namespace IronPDFSample
4.  {
5.      class Program
6.      {
7.          static void Main()
8.          {
9.              // Create a PDF from an existing HTML using C#
10.             var Renderer = new HtmlToPdf();
11.             var PDF = Renderer.RenderHTMLFileAsPdf("Assets/MyHTML.html");
12.             PDF.SaveAs("MyPdf.pdf");
13.         }
14.     }
15. }
```

4. Render Existing URL to PDF

Render existing web pages to PDFs in a few lines of C# or VB.Net code. Use this option when you need to convert a website that already has a well-formatted document to a PDF.

Call the `RenderHtmlAsPdf` to download web page content so that you can call `SaveAs` to export the content locally.

```
1. using IronPdf;
2.
3. namespace IronPDFSample
4. {
5.     class Program
6.     {
7.         static void Main()
8.         {
9.             // Create a PDF from any existing web page
10.            var Renderer = new HtmlToPdf();
11.            var PDF =
    Renderer.RenderUrlAsPdf("https://en.wikipedia.org/wiki/Portable_Document_Format");
12.            PDF.SaveAs("wikipedia.pdf");
13.        }
14.    }
15. }
```

5. ASP.NET Web Forms to PDF

Render ASP.NET web forms as PDF instead of HTML with a single line of code. Place the line of code in the `Page_Load` method of the page's code-behind.

- Create a new ASP.NET WebForms application or open an existing one
- Install the NuGet package
- Import the `IronPdf` namespace with the `using` keyword
- Open the code-behind for the page that you want to render to PDF. For example, `Default.aspx.cs`
- Call `RenderThisPageAsPdf` on AspxToPdf

```
using IronPdf;
using System;
using System.Web.UI;

namespace WebApplication7
{
    public partial class _Default : Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {           AspxToPdf.RenderThisPageAsPdf(AspxToPdf.FileBehavior.InBrowser);
        }
    }
}
```

6. Route ASP MVC View to PDF

Route the user to a PDF document with the ASP MVC framework. Use this option when creating a new ASP MVC app or add an existing MVC controller to an app.

Start the new project wizard in Visual Studio, and choose ASP.NET Web Application (.NET Framework) -> MVC. Or open an existing MVC project. Open the file `HomeController` in the Controllers folder and replace the Index method, or add a new controller.

This is an example of how the code should look:

```
using IronPdf;
using System;
using System.Web.Mvc;
namespace WebApplication8.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            var PDF = HtmlToPdf.StaticRenderUrlAsPdf(new Uri("https://en.wikipedia.org"));
            return File(PDF.BinaryData, "application/pdf", "Wiki.Pdf");
        }
        public ActionResult About()
        {
            ViewBag.Message = "Your application description page.";
            return View();
        }
        public ActionResult Contact()
        {
            ViewBag.Message = "Your contact page.";
            return View();
        }
    }
}
```

7. Add Headers and Footers

The PrintOptions property allows you to craft headers and footers for each page of the document. Access these options on the `HtmlToPdf` object. This sample works inside a .NET Core console app.

Use these template properties to build the content.

{page} {total-pages} {url} {date} {time} {html-title} & {pdf-title}

```
using IronPdf;

namespace ConsoleApp
{
    class Program
    {
        static void Main(string[] args)
        {
            var htmlToPdf = new HtmlToPdf();
            htmlToPdf.PrintOptions.FirstPageNumber = 1;
            //Header options
            htmlToPdf.PrintOptions.Header.DrawDividerLine = true;
            htmlToPdf.PrintOptions.Header.CenterText = "{url}";
            htmlToPdf.PrintOptions.Header.FontFamily = "Helvetica,Arial";
            htmlToPdf.PrintOptions.Header.FontSize = 12;
            //Footer options
            htmlToPdf.PrintOptions.Footer.DrawDividerLine = true;
            htmlToPdf.PrintOptions.Footer.FontFamily = "Arial";
            htmlToPdf.PrintOptions.Footer.FontSize = 10;
            htmlToPdf.PrintOptions.Footer.LeftText = "{date} {time}";
            htmlToPdf.PrintOptions.Footer.RightText = "{page} of {total-pages}";
            htmlToPdf.RenderHtmlAsPdf("<h1>Hello World<h1>").SaveAs("html-string.pdf");
        }
    }
}
```

7.1. Add Headers and Footers with HTML

As above, this sample works in a .NET Core console app. Specify HTML with the `HtmlFragment` property.

```
using IronPdf;
using System;
namespace ConsoleApp
{
    class Program
    {
        static void Main(string[] args)
        {
            var htmlToPdf = new HtmlToPdf();
            // Build a footer using html to style the text // mergeable fields are:
            // {page} {total-pages} {url} {date} {time} {html-title} & {pdf-title}
            htmlToPdf.PrintOptions.Footer = new HtmlHeaderFooter()
            {
                Height = 15,
                HtmlFragment = "<center><i>{page} of {total-pages}<i></center>",
                DrawDividerLine = true
            };
            // Build a header using an image asset
            // Note the use of BaseUrl to set a relative path to the assets
            htmlToPdf.PrintOptions.Header = new HtmlHeaderFooter()
            {
                Height = 20,
                HtmlFragment = "<img src='logo.jpg'>",
                BaseUrl = new Uri(@"C:\assets\images").AbsoluteUri
            };
            htmlToPdf.RenderHtmlAsPdf("<h1>Hello World<h1>").SaveAs("html-string.pdf");
        }
    }
}
```

8. Encrypt PDFs with a Password

Set the `Password` property of a PDF document to encrypt it and force the user to enter the correct password to view the document. This sample works in a .NET Core Console app

```
using IronPdf;

namespace ConsoleApp
{
    class Program
    {
        static void Main(string[] args)
        {
            var htmlToPdf = new HtmlToPdf();
            var pdfDocument = htmlToPdf.RenderHtmlAsPdf("<h1>Hello World<h1>");
            pdfDocument.Password = "strong!@#pass&^%word";
            pdfDocument.SaveAs("secured.pdf");
        }
    }
}
```

9. Merge and Split PDF Documents

Use the `Merge` method to merge multiple PDF documents together, or `CopyPages` to split a number of pages out of an existing document. Include PDFs in your project as Content to access them by filename.

```
using IronPdf;
using System.Collections.Generic;

namespace ConsoleApp
{
    class Program
    {
        static void Main(string[] args)
        {
            var htmlToPdf = new HtmlToPdf();
            //Join Multiple Existing PDFs into a single document
            var pdfDocuments = new List<PdfDocument>();
            pdfDocuments.Add(PdfDocument.FromFile("A.pdf"));
            pdfDocuments.Add(PdfDocument.FromFile("B.pdf"));
            pdfDocuments.Add(PdfDocument.FromFile("C.pdf"));

            var mergedPdfDocument = PdfDocument.Merge(pdfDocuments);
            mergedPdfDocument.SaveAs("merged.pdf");

            //Add a cover page
            mergedPdfDocument.PrependPdf(htmlToPdf.RenderHtmlAsPdf("<h1>Cover Page</h1><hr>"));
            //Remove the last page from the PDF and save again
            mergedPdfDocument.RemovePage(mergedPdfDocument.PageCount - 1);
            mergedPdfDocument.SaveAs("merged.pdf");

            //Copy pages 1,2 and save them as a new document.
            mergedPdfDocument.CopyPages(1, 2).SaveAs("exerpt.pdf");
        }
    }
}
```

10. Extract Images from PDF Documents

This feature requires an additional NuGet package. Install `System.Drawing.Common`.

Use the `ExtractAllText` to get text and the `ExtractAllImages` method to get images.

```
using IronPdf;

namespace ConsoleApp
{
    class Program
    {
        static void Main(string[] args)
        {
            var htmlToPdf = new HtmlToPdf();

            var pdfDocument = PdfDocument.FromFile("A.pdf");

            //Get all text
            var allText = pdfDocument.ExtractAllText();

            //Get all Images
            var allImages = pdfDocument.ExtractAllImages();

            //Or even find the images and text by page
            for (var index = 0; index < pdfDocument.PageCount; index++)
            {
                var pageNumber = index + 1;
                var pageText = pdfDocument.ExtractTextFromPage(index);
                var pageImages = pdfDocument.ExtractImagesFromPage(index);
            }
        }
    }
}
```

11. Enable JavaScript

```
using IronPdf;

namespace ConsoleApp
{
    class Program
    {
        static void Main(string[] args)
        {
            var htmlToPdf = new HtmlToPdf();
            htmlToPdf.PrintOptions = new PdfPrintOptions()
            {
                EnableJavaScript = true,
                RenderDelay = 100
            };
        }
    }
}
```

12. Use OCR Scanning

Use the IronOCR library to scan documents for visual text that is not plain text. You will need to install the Nuget package `IronOcr`. Learn more about scanning PDFs with IronOCR.

```
using IronOcr;
using IronPdf;

namespace ConsoleApp
{
    class Program
    {
        static void Main(string[] args)
        {
            var htmlToPdf = new HtmlToPdf();

            var advancedOcr = new AdvancedOcr()
            {
                CleanBackgroundNoise = false,
                ColorDepth = 4,
                ColorSpace = AdvancedOcr.OcrColorSpace.Color,
                EnhanceContrast = false,
                DetectWhiteTextOnDarkBackgrounds = false,
                RotateAndStraighten = false,
                Language = IronOcr.Languages.English.OcrLanguagePack,
                EnhanceResolution = false,
                InputImageType = AdvancedOcr.InputTypes.Document,
                ReadBarCodes = true,
                Strategy = AdvancedOcr.OcrStrategy.Fast
            };

            var results = advancedOcr.ReadPdf(@"C:\Users\Me\Desktop\Invoice.pdf");
            var pages = results.Pages;
            var barcodes = results.Barcodes;
            var text = results.Text;
        }
    }
}
```

13. Use More Printing Options

Here are some more detailed printing options

```
using IronPdf;
using System.Text;

namespace ConsoleApp
{
    class Program
    {
        static void Main(string[] args)
        {
            var htmlToPdf = new HtmlToPdf();

            htmlToPdf.PrintOptions.SetCustomPaperSizeInInches(12.5, 20);
            htmlToPdf.PrintOptions.PrintHtmlBackgrounds = true;
            htmlToPdf.PrintOptions.PaperOrientation =
PdfPrintOptions.PdfPaperOrientation.Portrait;
            htmlToPdf.PrintOptions.Title = "My PDF Document Name";
            htmlToPdf.PrintOptions.EnableJavaScript = true;
            htmlToPdf.PrintOptions.RenderDelay = 50;
            htmlToPdf.PrintOptions.CssMediaType = PdfPrintOptions.PdfCssMediaType.Screen;
            htmlToPdf.PrintOptions.DPI = 300;
            htmlToPdf.PrintOptions.FitToPaperWidth = true;
            htmlToPdf.PrintOptions.JpegQuality = 80;
            htmlToPdf.PrintOptions.GrayScale = false;
            htmlToPdf.PrintOptions.FitToPaperWidth = true;
            htmlToPdf.PrintOptions.InputEncoding = Encoding.UTF8;
            htmlToPdf.PrintOptions.Zoom = 100;
            htmlToPdf.PrintOptions.CreatePdfFormsFromHtml = true;
            htmlToPdf.PrintOptions.MarginTop = 40;
            //millimenters
            htmlToPdf.PrintOptions.MarginLeft = 20;
            //millimenters
            htmlToPdf.PrintOptions.MarginRight = 20;
            //millimenters
            htmlToPdf.PrintOptions.MarginBottom = 40;
            //millimenters
            htmlToPdf.PrintOptions.FirstPageNumber = 1;
            //use 2 if a cover page will be appended
            htmlToPdf.RenderHTMLFileAsPdf("my-content.html").SaveAs("my-content.pdf");
        }
    }
}
```

14. Download the C# PDF Cheat Sheet

We have compiled this tutorial as an easy to read and share PDF document that explains in full how to create and edit PDFs in C# and VB.Net using the IronPDF library.

You can download it and use it as a develop guide for your .NET projects, or print it as a handy companion for IronPDF development. This saves time and effort in getting started adding PDF features to any .NET project.

15. Learn More

To learn more about HTML to PDF in C# or VB.Net applications, please read the detailed C# HTML to PDF Tutorial. The tutorial clearly explains advanced PDF settings with HTML templates, CSS, Images, and Javascript.

If you're interested in how to dynamically render ASPX pages in ASP.NET applications as PDFs, check out the full ASPX to PDF Tutorial.

A full IronPDF object reference for .NET developers is also available.