**How to create a PDF file in Blazor using C#**

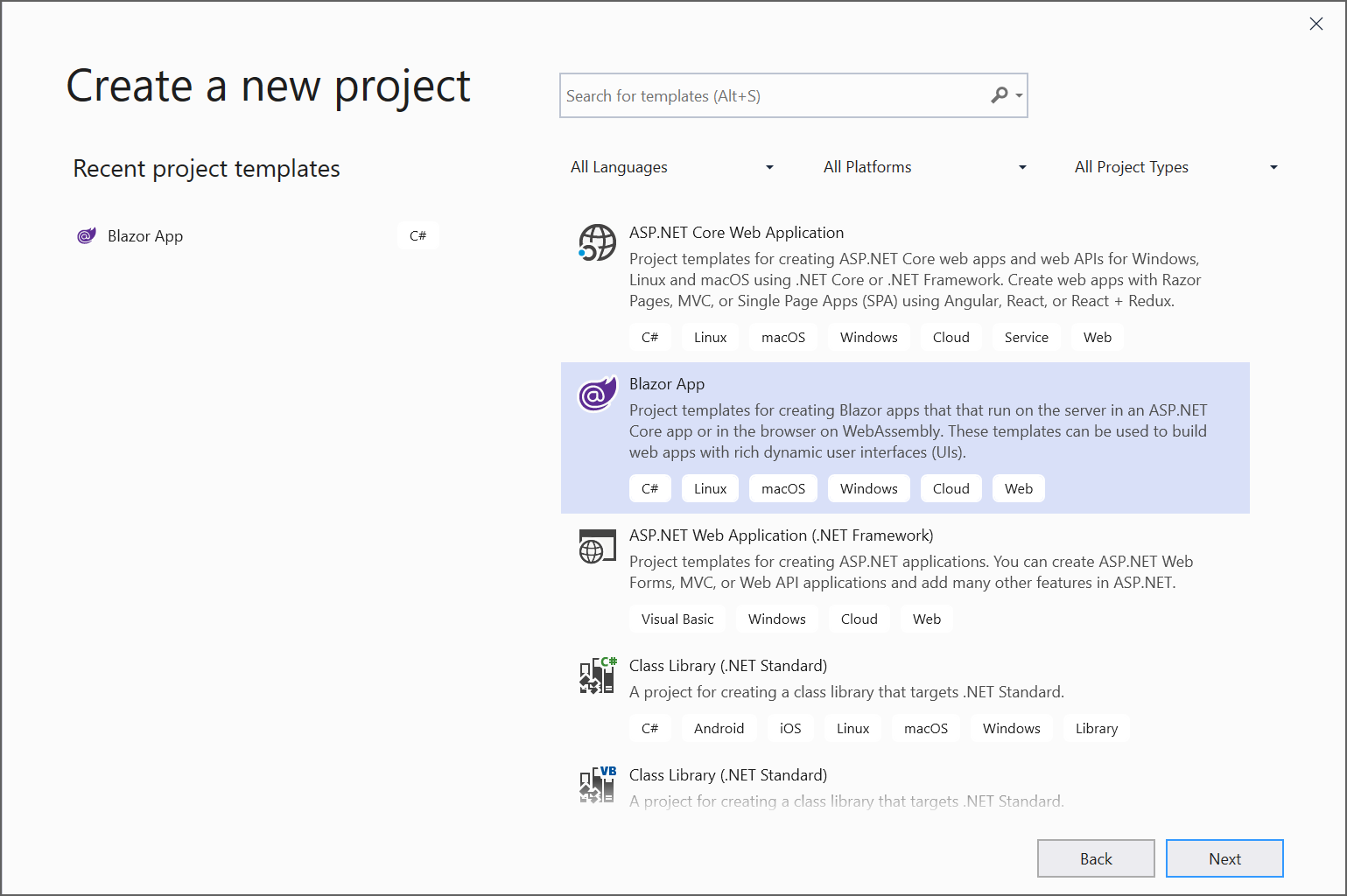
Creating a PDF document from the scratch using [**Syncfusion .NET Core library**](https://www.syncfusion.com/document-processing/pdf-framework/net-core/pdf-library) is easy. This article explains how to create a PDF file in Blazor framework using C#.

Prerequisites

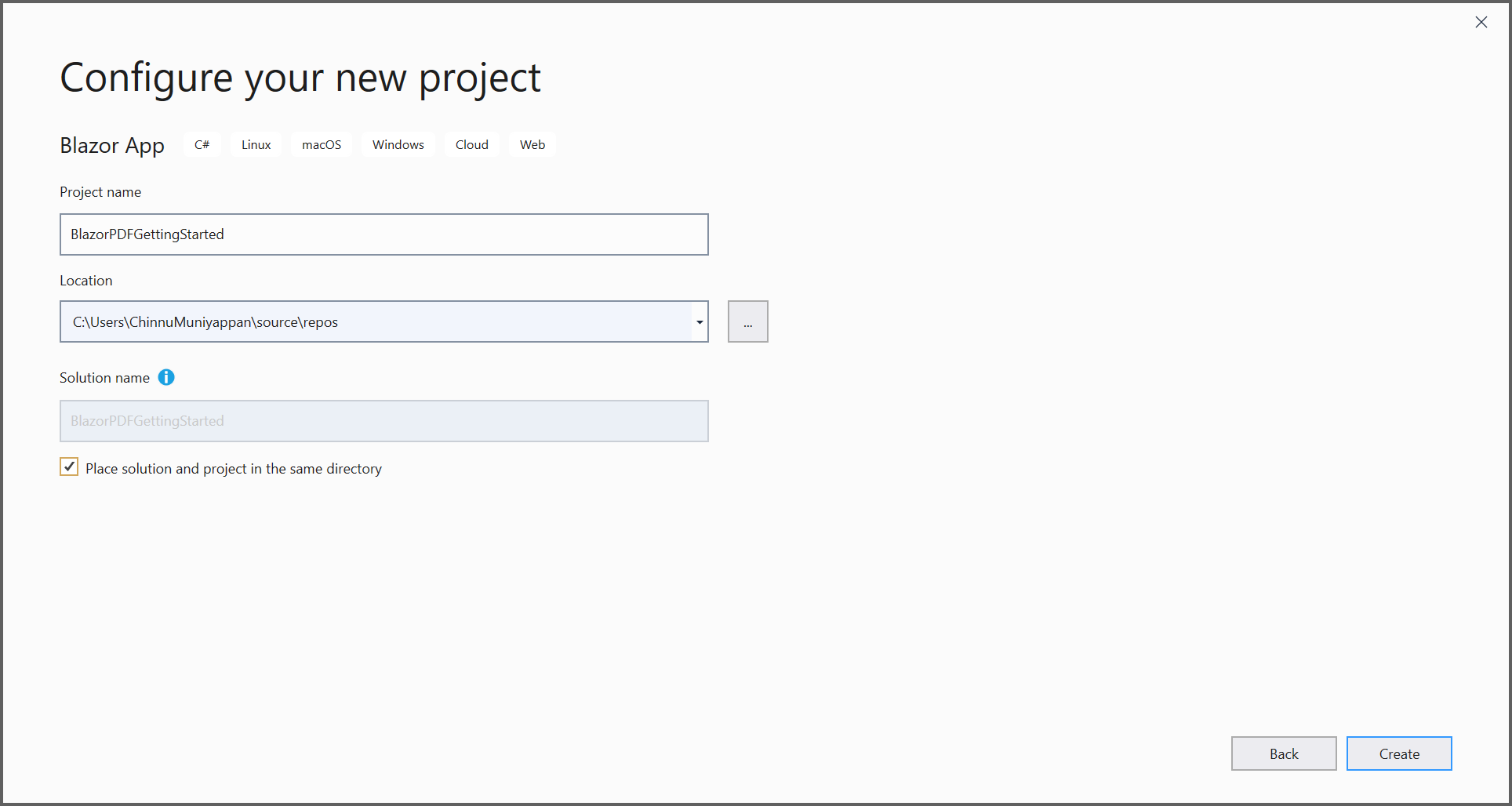
* + Visual Studio 2019
  + Install .NET Core SDK 6.0 ([**https://dotnet.microsoft.com/en-us/download/dotnet/6.0**](https://dotnet.microsoft.com/en-us/download/dotnet/6.0))

Creating a Blazor project

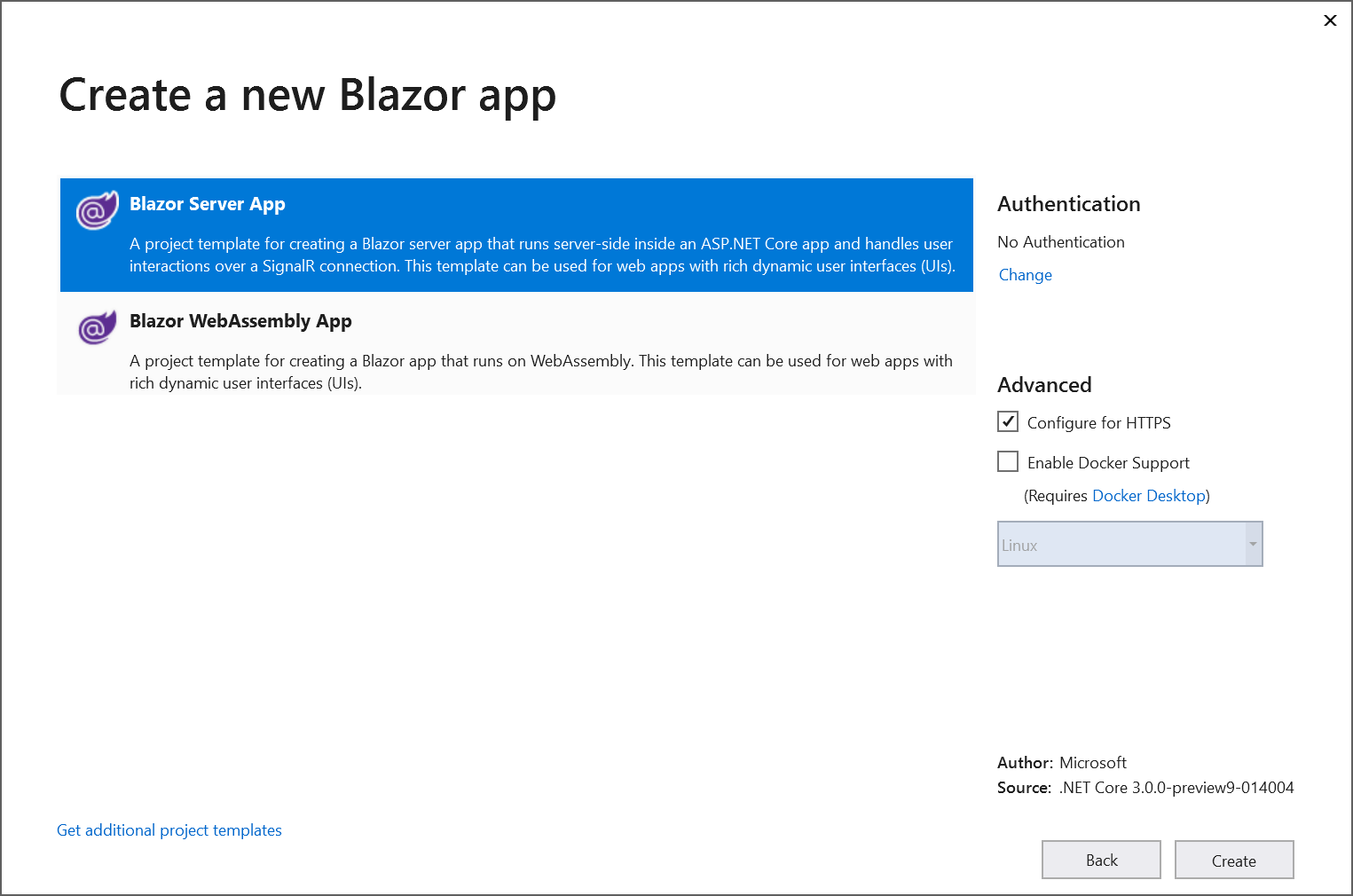
* Enable Visual Studio to use preview SDKs:
* Open Tools > Options in the menu bar.
* Open the **Projects and Solutions** node. Open the .NET Core tab.
* Check the box for **Use previews of the .NET Core SDK**. Select OK.
* Restart the Visual Studio 2019.
* Create a new project



* Select **Blazor App**. Select Next.

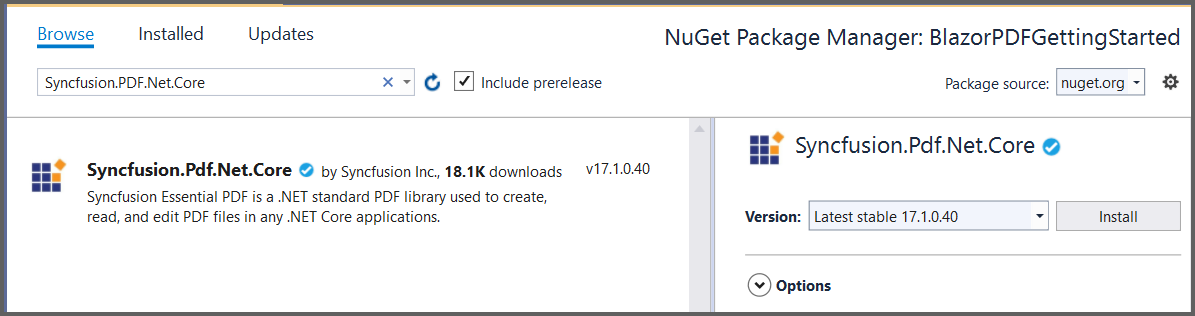


* Select **Blazor Server** **APP**.



Creating a PDF file in Blazor

* To create a PDF file, install **[Syncfusion.PDF.Net.Core](https://www.nuget.org/packages/Syncfusion.Pdf.Net.Core/" \t "_blank)** to the Blazor project.



* Add the following namespace in the Index.razor to create a PDF document from the scratch.
* @using Syncfusion.Pdf;
* @using Syncfusion.Pdf.Graphics;
* @using System.IO;
* @inject Microsoft.JSInterop.IJSRuntime JS
* Add a button and hook the click event function.
* <**button** class="btn btn-primary" @**onclick**="@CreatePDF">Create PDF</**button**>
* Add the following code to create a PDF file in Blazor.
* @functions {
* void CreatePDF()
* {
* //Create a new PDF document
* PdfDocument document = **new** PdfDocument();
* //Add a page to the document
* PdfPage page = document.Pages.Add();
* //Create PDF graphics for the page
* PdfGraphics graphics = page.Graphics;
* //Set the standard font
* PdfFont font = **new** PdfStandardFont(PdfFontFamily.Helvetica, 20);
* //Draw the text
* graphics.DrawString("Hello World!", font, PdfBrushes.Black, new Syncfusion.Drawing.PointF(0, 0));
* //Saving the PDF to the MemoryStream
* MemoryStream stream = **new** MemoryStream();
* document.Save(stream);
* document.Close(true);

* //Download the PDF in the browser.
* }
* }

To save the pdf document in the project folder:

* Add the following code snippet in the created class file to save the pdf document in the project folder itself:
* System.IO.Stream output;
* output = System.IO.File.OpenWrite("output.pdf");
* //Save the document.
* doc.Save(output);
* //Close the document.
* doc.Close(true);
* output.Close();

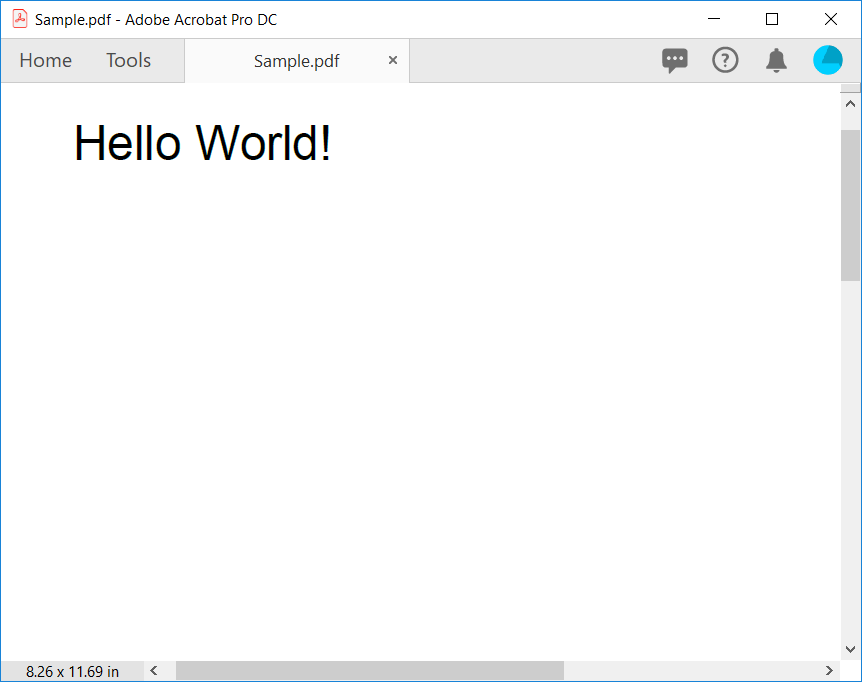
To download the PDF file in browser

Create a class file with FileUtil name and add the following code to invoke the JavaScript action to download the file in the browser.

* Add the following code in the created class file.
* public static **class** FileUtil
* {
* public static ValueTask<**object**> SaveAs(this IJSRuntime js, string filename, byte[] data)
* => js.InvokeAsync<**object**>(
* "saveAsFile",
* filename,
* Convert.ToBase64String(data));
* }
* Add the following JavaScript function in the \_Host.cshtml in the pages folder.
* <script **type**="text/javascript">
* **function** saveAsFile(filename, bytesBase64) {
* **if** (navigator.msSaveBlob) {
* //Download document in Edge browser
* var data = window.atob(bytesBase64);
* var bytes = **new** Uint8Array(data.length);
* **for** (var i = 0; i < data.length; i++) {
* bytes[i] = data.charCodeAt(i);
* }
* var blob = **new** Blob([bytes.buffer], { type: "application/octet-stream" });
* navigator.msSaveBlob(blob, filename);
* }
* **else** {
* var link = document.createElement('a');
* link.download = filename;
* link.href = "data:application/octet-stream;base64," + bytesBase64;
* document.body.appendChild(link); // Needed for Firefox
* link.click();
* document.body.removeChild(link);
* }
* }
* </script>
* Finally, add the following code in the Index.razor.
* //Download the PDF in the browser.
* **JS**.SaveAs("Sample.pdf", stream.ToArray());
* The complete code of the PDF creation will look like below.
* @functions {
* void CreatePDF()
* {
* //Create a new PDF document
* PdfDocument document = **new** PdfDocument();
* //Add a page to the document
* PdfPage page = document.Pages.Add();
* //Create PDF graphics for the page
* PdfGraphics graphics = page.Graphics;
* //Set the standard font
* PdfFont font = **new** PdfStandardFont(PdfFontFamily.Helvetica, 20);
* //Draw the text
* graphics.DrawString("Hello World!", font, PdfBrushes.Black, new Syncfusion.Drawing.PointF(0, 0));
* //Saving the PDF to the MemoryStream
* MemoryStream stream = **new** MemoryStream();
* document.Save(stream);
* document.Close(true);
* //Download the PDF in the browser.
* JS.SaveAs("Sample.pdf", stream.ToArray());
* }
* }

You can download the sample from [**BlazorPDFGettingStarted.zip**](https://www.syncfusion.com/downloads/support/directtrac/general/ze/BlazorPDFGettingStarted-1007429454)

By executing the program, you will get the PDF file as follows.



Take a moment to peruse the [**documentation**](https://help.syncfusion.com/file-formats/pdf/working-with-text), where you can find other options like drawing right-to-left text and multi-column text, consuming TrueType fonts, Standard fonts, and CJK fonts. Also, the features like [**PDF form filling**](https://help.syncfusion.com/file-formats/pdf/working-with-forms), convert [**HTML to PDF**](https://help.syncfusion.com/file-formats/pdf/convert-html-to-pdf/webkit) , and [**protect PDF documents**](https://help.syncfusion.com/file-formats/pdf/working-with-security) with code examples.

Refer [**here**](https://www.syncfusion.com/document-processing/pdf-framework/net-core) to explore the rich set of Syncfusion Essential® PDF features.

**Conclusion**

I hope you enjoyed learning about how to create a PDF file in Blazor using C#.

You can refer to our [**.NET Core PDF**](https://www.syncfusion.com/document-processing/pdf-framework/net-core) [**feature tour**](https://www.syncfusion.com/document-processing/pdf-framework/net-core) page to know about its other groundbreaking feature representations and [**documentation**](https://help.syncfusion.com/file-formats/pdf/create-pdf-file-in-asp-net-core), and how to quickly get started for configuration specifications. You can also explore our [**.NET Core PDF**](https://ej2.syncfusion.com/aspnetcore/PDF/Default?_gl=1*oxrtb5*_ga*MjkzODA3NDIuMTY4MjQwOTYyOA..*_ga_WC4JKKPHH0*MTY4ODQ0ODEzOS4xODQuMS4xNjg4NDUwNjc2LjI5LjAuMA..*_ga_2QTHE2Y2YX*MTY4ODQ0ODEzOS44LjEuMTY4ODQ1MDY3Ni4yOS4wLjA.&_ga=2.48642854.1072980492.1688360769-29380742.1682409628#/bootstrap5) [**example**](https://ej2.syncfusion.com/aspnetcore/PDF/Default#/bootstrap5) to understand how to create and manipulate data.

For current customers, you can check out our components from the [**License and Downloads**](https://www.syncfusion.com/sales/teamlicense) page. If you are new to Syncfusion®, you can try our 30-day [**free trial**](https://www.syncfusion.com/downloads/aspnetcore-js2)to check out our other controls.

If you have any queries or require clarifications, please let us know in the comments section below. You can also contact us through our [**support forums**](https://www.syncfusion.com/forums), [**Direct-Trac**](https://support.syncfusion.com/create), or [**feedback portal**](https://www.syncfusion.com/feedback/aspnet-core?control=pdf). We are always happy to assist you!