# Creating PDF document in .NET Core Web API

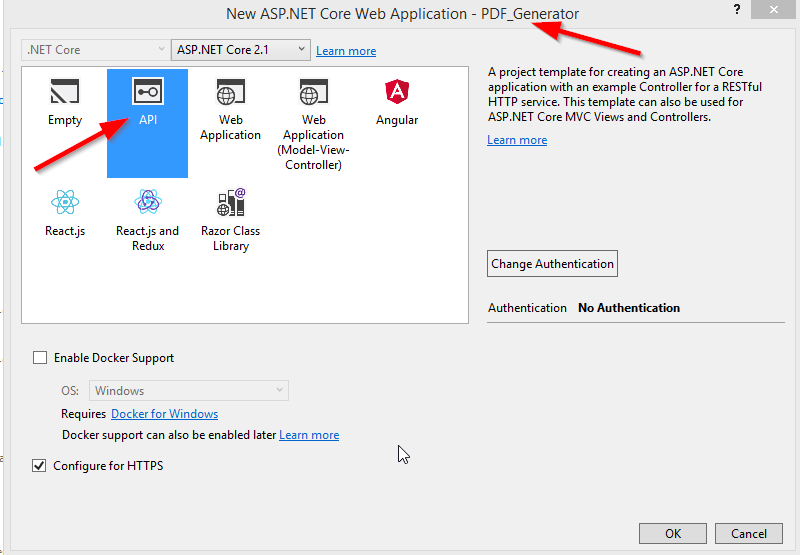
Let’s imagine that we have a .NET Core Web API project in which we need to generate a PDF report. Even though it shouldn’t suppose to be too hard to do something like that, we could end up in loosing to much time if we don’t know how to do it properly.

In this article, we are going to show you how to use DinkToPDF library to easily generate PDF documents while working on the .NET Core Web API project.

So, without further ado, let’s dive right into the fun part.

## Basic Project Preparations

Let’s start from the very beginning, by creating a new .NET Core Web Api project named PDF\_Generator:



After the project creation, let’s modify the launchSettings.json file to disable our browser to start automatically and to change the applicationUrl property to localhost:5000:

{

"$schema": "http://json.schemastore.org/launchsettings.json",

"iisSettings": {

"windowsAuthentication": false,

"anonymousAuthentication": true,

"iisExpress": {

"applicationUrl": "http://[::]:5000",

"sslPort": 44320

}

},

"profiles": {

"IIS Express": {

"commandName": "IISExpress",

"launchBrowser": false,

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development"

}

},

"PDF\_Generator": {

"commandName": "Project",

"launchBrowser": false,

"applicationUrl": "http://[::]:5000",

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development"

}

}

}

}

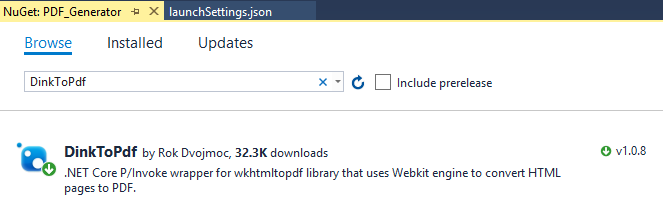
## DinkToPdf Library Configuration

DinkToPdf is a cross-platform oriented library which is the wrapper for the Webkit HTML to PDF library. It uses WebKit engine to convert HTML to PDF.

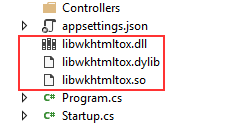
So, let’s first install the DinkToPdf library:

PM> Install-Package DinkToPdf

Or search for DinkToPdf inside the Nuget Package Window:



After the installation completes, we need to import native library files to our root project. We can find those files in our source project in the NativeLibrary folder. Inside we will find two folders 32bit and 64bit, so we need to choose the appropriate library for our OS. We are going to choose the files from the 64bit folder:



Finally, we need to register this library with our IoC container in the StartUp class:

public void ConfigureServices(IServiceCollection services)

{

services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version\_2\_1);

services.AddSingleton(typeof(IConverter), new SynchronizedConverter(new PdfTools()));

}

Excellent.

We have everything in place and we are ready for the coding part.