# How to export HTML to PDF in ASP.NET Core

## Introduction

Many third party libraries can do this job in .NET Core, but they all need to buy a licence.

This sample demonstrates how to export HTML to PDF using node.js in ASP.NET Core.

## Sample prerequisites

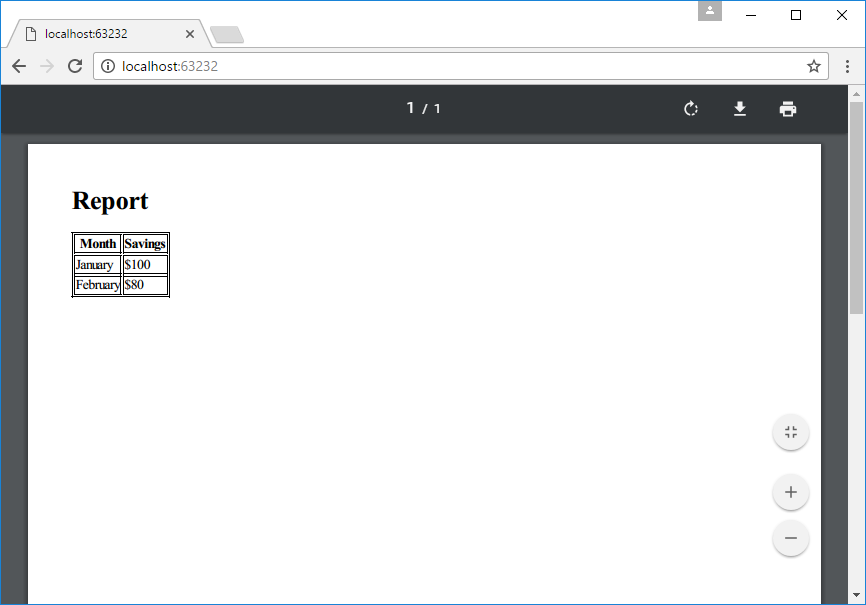
* Visual Studio 2017 or above. [[Visual Studio Home Page](https://www.visualstudio.com/)]
* Visual Studio enabled ASP.NET Core develop component.
* Node.js 6.10.2 or later. [[Node.js and Npm](https://nodejs.org/en/)]
* Npm 3.0 or above.

## Building the sample

Use Visual Studio to open the sample solution, then press **F6 Key** to build the sample project.

## Running the sample

* Use Visual Studio to open the sample solution, then press F5 Key or select **Debug -> Start Debugging** from the menu.
* When the project is running, you will see below page in browser.



## Using the code

Package.json of node.js

{

"name": "pdf",

"version": "1.0.0",

"description": "",

"main": "index.js",

"dependencies": {

"jsreport-core": "^1.3.1",

"jsreport-phantom-pdf": "^1.4.4",

"jsreport-jsrender": "^1.0.2"

},

"devDependencies": {},

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"author": "",

"license": "ISC"

}

Generate PDF function with node.js, file name is pdf.js

module.exports = function (callback, html) {

var jsreport = require('jsreport-core')();

jsreport.init().then(function () {

return jsreport.render({

template: {

content: html,

engine: 'jsrender',

recipe: 'phantom-pdf'

}

}).then(function (resp) {

callback(null, resp.content.toJSON().data);

});

}).catch(function (e) {

callback(e, null);

})

};

Source HTML

<!DOCTYPE html>

<html>

<head>

<style>

table, th, td { border: 1px solid black; }

</style>

</head>

<body>

<h1>Report</h1>

<table>

<tr>

<th>Month</th>

<th>Savings</th>

</tr>

<tr>

<td>January</td>

<td>$100</td>

</tr>

<tr>

<td>February</td>

<td>$80</td>

</tr>

</table>

</body>

</html>

Startup.cs

public class Startup

{

public void ConfigureServices(IServiceCollection services)

{

services.AddNodeServices();// this is in package Microsoft.AspNetCore.NodeServices

services.AddMvc();

}

public void Configure(IApplicationBuilder app)

{

app.UseStaticFiles();

app.UseMvcWithDefaultRoute();

}

}

Controller

[HttpGet]

public async Task<IActionResult> Index([FromServices] INodeServices nodeServices)

{

HttpClient hc = new HttpClient();

var htmlContent = await hc.GetStringAsync($"http://{Request.Host}/report.html");

var result = await nodeServices.InvokeAsync<byte[]>("./pdf", htmlContent);

HttpContext.Response.ContentType = "application/pdf";

HttpContext.Response.Headers.Add("x-filename", "report.pdf");

HttpContext.Response.Headers.Add("Access-Control-Expose-Headers", "x-filename");

HttpContext.Response.Body.Write(result, 0, result.Length);

return new ContentResult();

}

## For building new project

1. Copy package.json to new project root.

Main dependencies are **jsreport-core**, **jsreport-phantom-pdf**, **jsreport-jsrender**.

1. Add file pdf.js to project root.
2. Add nuget package **Microsoft.AspNetCore.NodeServices**.
3. Add below code in Startup/ConfigureServices

services.AddNodeServices();

1. Use node service in controller. See file HomeController.cs

## More information

Introducing jsreport-core

<https://jsreport.net/blog/introducing-jsreport-core>

About NodeServices

<http://blog.nbellocam.me/2016/08/24/javascriptservices-asp-net-core/>