Readme - C# SDK

Welcome to FusionExport

First of all, thank you very much for downloading FusionExport. Initially, we developed FusionExport to help our customers to generate charts on servers. Since the inception, we have improved FusionExport to help you export not only a single chart but also dashboards.

Now that you have downloaded FusionExport let's get started!

Getting started

Installation of FusionExport server

Look for the `fusionexport` launcher in the downloaded package and start the server by running following command in Terminal:

```
./fusionexport
```

If you are using a Windows environment then run the following command in PowerShell

```
./fusionexport.bat
```

Once you execute the command successfully, FusionExport server will start on the port 1337. You will see console logs similar to the following:

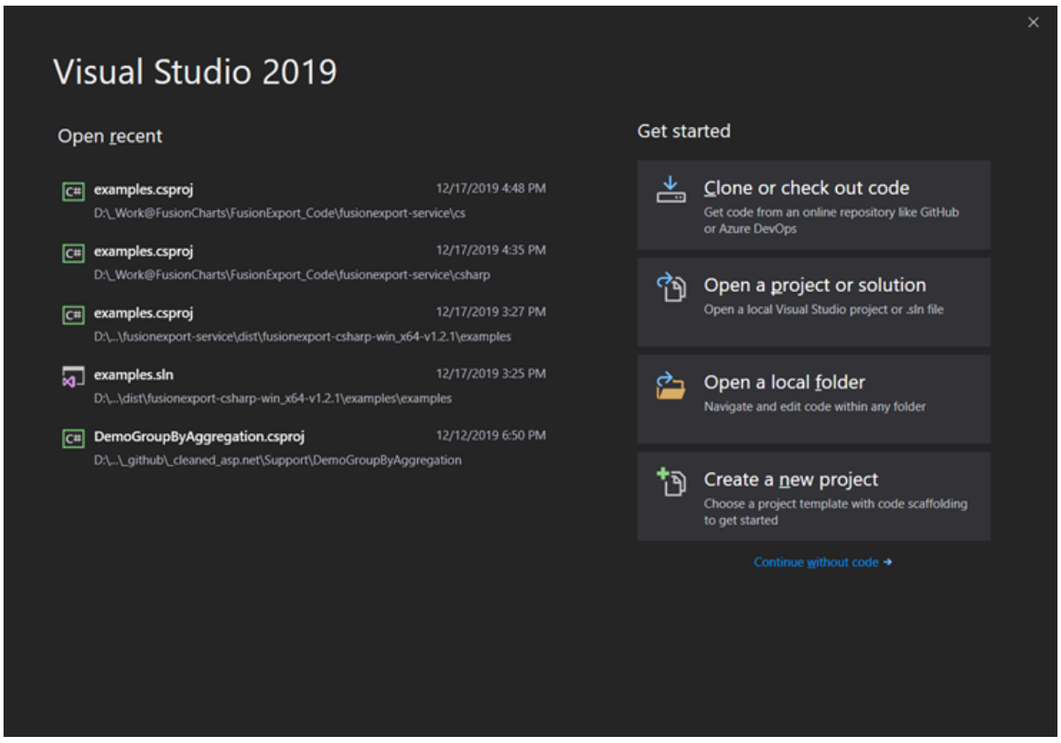
```
[2019-12-06T14:41:43] [debug] [WorkerPoolManager] Task Create-Pool: Creating worker pool
[2019-12-06T14:41:43] [debug] [WorkerPoolManager] Booting chrome worker
[2019-12-06T14:41:43] [debug] [WorkerFactory] Creating worker
[2019-12-06T14:41:43] [debug] [WorkerPoolManager] Task Create-Pool: Creating worker pool
[2019-12-06T14:41:43] [debug] [WorkerPoolManager] Booting chrome worker
[2019-12-06T14:41:43] [debug] [WorkerFactory] Creating worker
[2019-12-06T14:41:44] [debug] [ExporterFactory] Initializing component
[2019-12-06T14:41:44] [debug] [ChromeWorker] Initializing component
[2019-12-06T14:41:44] [debug] [ExporterFactory] Initializing component
[2019-12-06T14:41:44] [debug] [ChromeWorker] Initializing component
[2019-12-06T14:41:44] [info] [Handler] Listening on 127.0.0.1:1337
[2019-12-06T14:41:44] [info] [Handler] Listening on 127.0.0.1:1337
```

> In case if you face any issues while starting the server, get in touch with our support team on support@fusioncharts.com to resolve the issue.

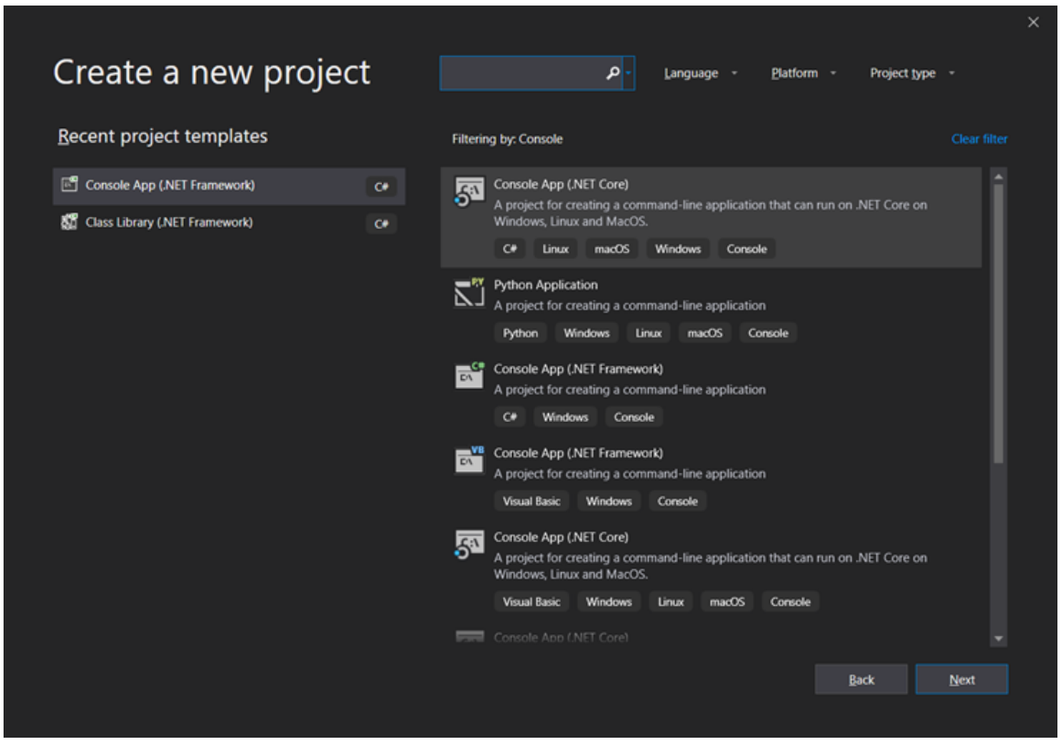
Create a sample project

Now that our server is running let's create a sample project in C# using Visual Studio.

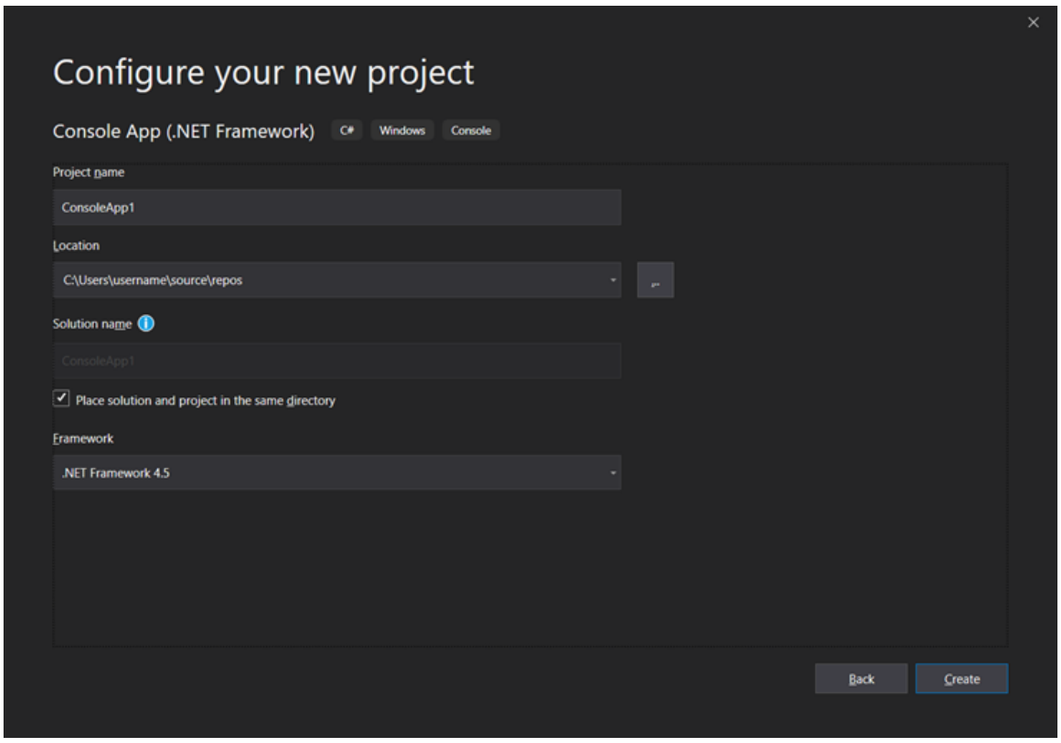
To create a sample project in C#, open Visual Studio and follow the steps below:



Click on 'Create a new project'

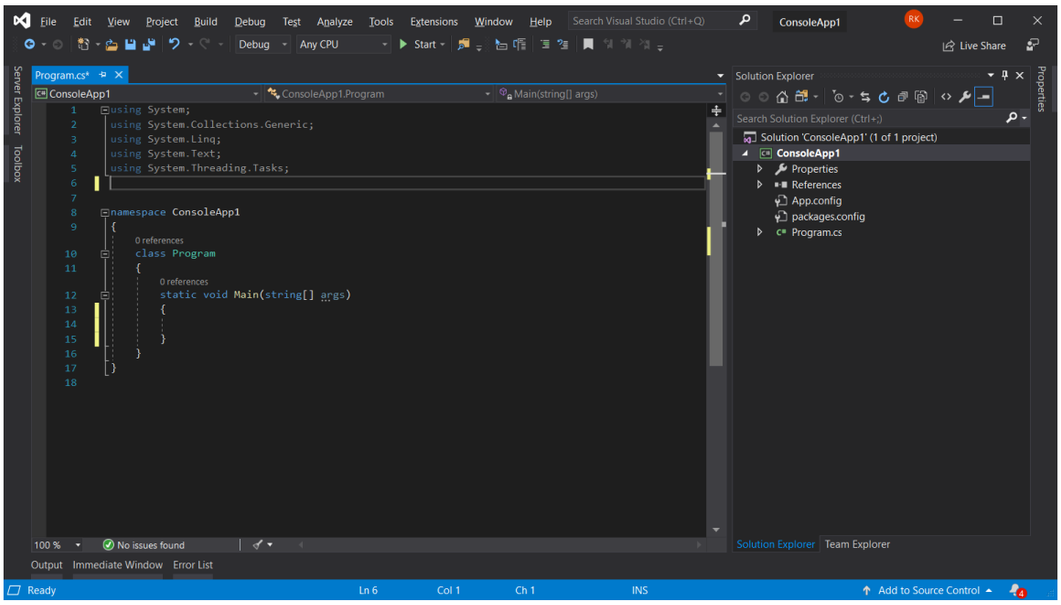


Click on ‘Console App (.NET Framework)’



Configure the project name, location etc and click on ‘Create’

Once project is created, it will look like this:



Install FusionExport C# client

Now that we have created a project, let’s install FusionExport C# client from the Nuget. To do that, open the Nuget Package Manager Console and type the following command:

```
Install-Package FusionExport
```

Import dependencies

Double click on `Program.cs` file in the Solution Explorer, type `using FusionCharts.FusionExport.Client;` then go to the method `Main` and instantiate `ExportConfig` class from `FusionCharts.FusionExport.Client`. This modules will help us perform export related operations and set configurations, respectively. You can read more about it on the API reference.

```
using FusionCharts.FusionExport.Client;

namespace ConsoleApp1
{
    class Program
    {
        static void Main(string[] args)
        {
            // Instantiate the ExportConfig class and add the required configurations
            ExportConfig exportConfig = new ExportConfig();
        }
    }
}
```

Define FusionCharts configurations

Now that we have imported FusionExport in your project, let's add a simple chart configuration which we will export. FusionExport, by default, accepts FusionCharts configurations to render charts (however, we also support almost all the third party JavaScript charting libraries as well). For this example, we will be exporting a simple column chart:

```
using System;
using System.Collections.Generic;
using FusionCharts.FusionExport.Client;

namespace ConsoleApp1
{
    class Program
    {
        static void Main(string[] args)
        {
            // Instantiate the ExportConfig class and add the required configurations
            ExportConfig exportConfig = new ExportConfig();
            List<string> results = new List<string>();

            string chartConfig = @"{
                        ""type"": ""column2d"",
                        ""renderAt"": ""chart-container"",
                        ""width"": ""600"",
                        ""height"": ""400"",
                        ""dataFormat"": ""json"",
                        ""dataSource"": {
                            ""chart"": {
                                ""caption"": ""Number of visitors last week"",
                                ""subCaption"": ""Bakersfield Central vs Los Angeles Topanga""
                            },
                            ""data"": [{
                                    ""label"": ""Mon"",
                                    ""value"": ""15123""
                                },{
                                    ""label"": ""Tue"",
                                    ""value"": ""14233""
                                },{
                                    ""label"": ""Wed"",
                                    ""value"": ""25507""
                                }
                            ]
                        }
                    }";

            // Instantiate the ExportManager class
            using (ExportManager exportManager = new ExportManager())
            {
                exportConfig.Set("chartConfig", chartConfig);
```

As you can see above, we have defined a simple chart and assigned it to `chartConfig` parameter of `Set` method of `exportConfig` object. There are many properties that you can set in `exportConfig`. You can get the full list on the API Reference.

In case if you are curious about FusionCharts and various charts configurations, go through the chart attributes on Dev Center.

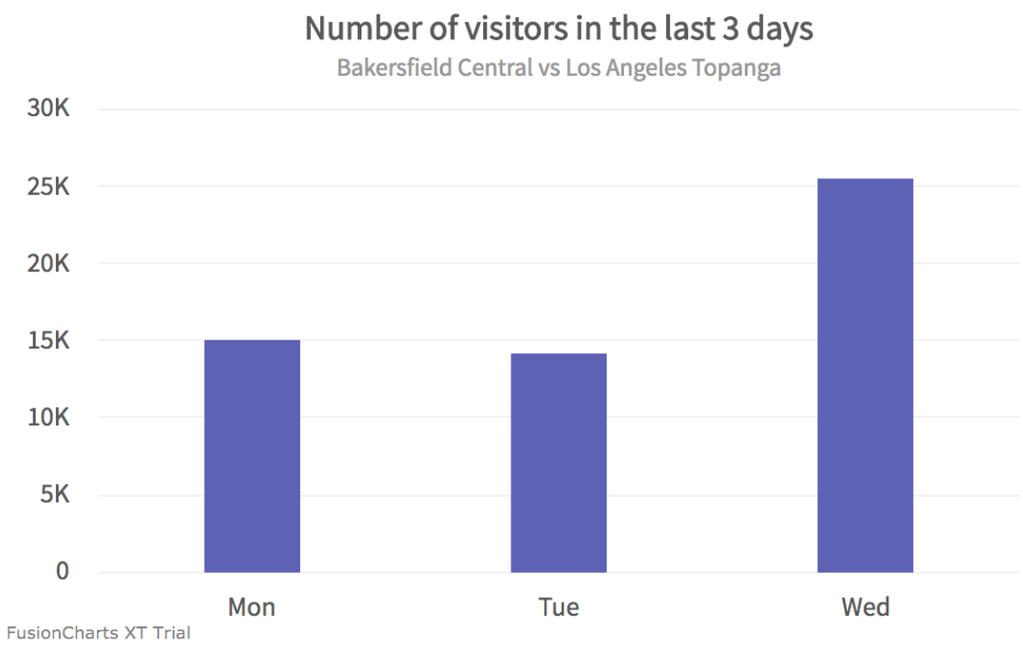
Let's export

It is time to export our chart. First, let's define `type` to PDF (default type is `PNG`) and export the chart:

```
using System;
using System.Collections.Generic;
using FusionCharts.FusionExport.Client;

namespace ConsoleApp1
{
    class Program
    {
        static void Main(string[] args)
        {
            // Instantiate the ExportConfig class and add the required configurations
            ExportConfig exportConfig = new ExportConfig();
            List<string> results = new List<string>();

            string chartConfig = @"{
                        ""type"": ""column2d"",
                        ""renderAt"": ""chart-container"",
                        ""width"": ""600"",
                        ""height"": ""400"",
                        ""dataFormat"": ""json"",
                        ""dataSource"": {
                            ""chart"": {
                                ""caption"": ""Number of visitors last week"",
                                ""subCaption"": ""Bakersfield Central vs Los Angeles Topanga""
                            },
                            ""data"": [{
                                    ""label"": ""Mon"",
                                    ""value"": ""15123""
                                },{
                                    ""label"": ""Tue"",
                                    ""value"": ""14233""
                                },{
                                    ""label"": ""Wed"",
                                    ""value"": ""25507""
                                }
                            ]
                        }
                    }";

            // Instantiate the ExportManager class
            using (ExportManager exportManager = new ExportManager())
            {
                exportConfig.Set("chartConfig", chartConfig);
                exportConfig.Set("type", "pdf");
                string tmpPath = Environment.GetEnvironmentVariable("%TMP%", EnvironmentVariableTarget.User);
                // Call the Export() method with the export config
                results.AddRange(exportManager.Export(exportConfig, tmpPath, true));
            }

            foreach (string path in results)
            {
                Console.WriteLine(path);
            }

            Console.Read();

        }
    }
}
```

By calling `Export` method, you have started the export against the chart you defined in `exportConfig`. `Export` method returns a `List<string>` which will give you an array of the exported file path. In this case, only one export file will be generated with the default name `export.pdf` in the same directory. To run the sample, press `F5`.

Once you have exported, your exported file will look like this:



Congratulations! You have successfully exported a chart using FusionExport.

More examples

For your convenience, we have already created a few samples and hosted them on Github. It will help you to get started with FusionExport. To run these samples, you'll have to clone the repository:

```
git clone https://github.com/fusioncharts/fusionexport-csharp-examples.git
```

Once the repository is cloned, you will get the project and solution files along with the samples.

> **Important Note:** All the examples use FusionCharts to render charts. In case if you want to use any other JavaScript charting libraries, then run samples from `using_other_libraries` directory.

Before you start executing these examples, we first need to set up the NuGet packages. This is an important step. It will restore the FusionExport C# NuGet package from the NuGet. To do this, open the `examples.csproj` from the repository folder that you have just cloned, click on `Tools > NuGet Package Manager > Package Manager Console` and run the following command:

```
Update-Package -reinstall
```

Export multiple charts

Since we already explained above how to export your first chart, let's export multiple charts. We have written a thorough guide on how to export multiple charts in Dev Center. For the

simplicity, open the `examples.csproj` in Visual Studio. Open the `Program.cs` from Solution Explorer and use the following code:

```
Examples.example_2_export_multiple_charts.Run();
```

Once you have made the above changes to the `Program.cs` file, `press F5` to run the project and export the chart.

Export a dashboard

We have already written an in-depth tutorial to export your dashboard in Dev Center. However, we have provided a sample code in the C# project, please add the following line in `Program.cs` and run it:

```
Examples.example_3_export_dashboard.Run();
```

Upon successful execution, you will get a file `export.pdf`, which will consist of the dashboard you just exported.

Send dashboard via email

You can also use FusionExport to send dashboards as an attachment. For this example, we are using .NET Framework's native `SmtpClient` to send emails. Open `Program.cs` and add the following line:

```
Examples.example_4_send_email_report.Run();
```

Now, open example_4_send_email_report.cs file from Solution Explorer and change the following parameters:

1. `<SENDER'S EMAIL>`: Email address of the sender
2. `<RECEIVERS'S EMAIL>`: Email address of the recipient
3. `<HOST>`: SMTP server address (for example - smtp.gmail.com)
4. `<USERNAME>`: User name for the SMTP authentication
5. `<PASSWORD>`: Password for the SMTP authentication

Based on the email server you have, you may have to modify parameters like `Port`, `EnableSsl`, etc.

Once you have provided this parameters, run the example by `pressing F5.`

Upon successful, execution, you will get an email from the details you have provided.

Add headers and footers

You can also add headers and footers in the exported PDF document by modifying properties like `headerComponents` and `footerComponents`. Add the following line in `Program.cs` and run the project.

```
Examples.example_5_set_header_footer.Run();
```

More Resources

- Documentation

- API Reference
  - FusionExport C# SDK
  - FusionExport Server
- Public Roadmap

In case if you have any feedback or if you are stuck anywhere, reach out to product@fusioncharts.com.