

RTF to HTML .Net

(Multi-platform .Net library)

[SautinSoft](#)

Linux development manual

Table of Contents

1. Preparing environment	2
1.1. Check the installed Fonts availability	3
2. Creating "Convert RTF/DOCX to HTML" app.....	5

1. Preparing environment

In order to build multi-platform applications using .NET on Linux, the first steps are for installing in our Linux machine the required tools.

We need to install .NET SDK from Microsoft and to allow us to develop easier, we will install an advance editor with a lot of features, Visual Studio Code from Microsoft.

Both installations are very easy and the detailed description can be found by these two links:

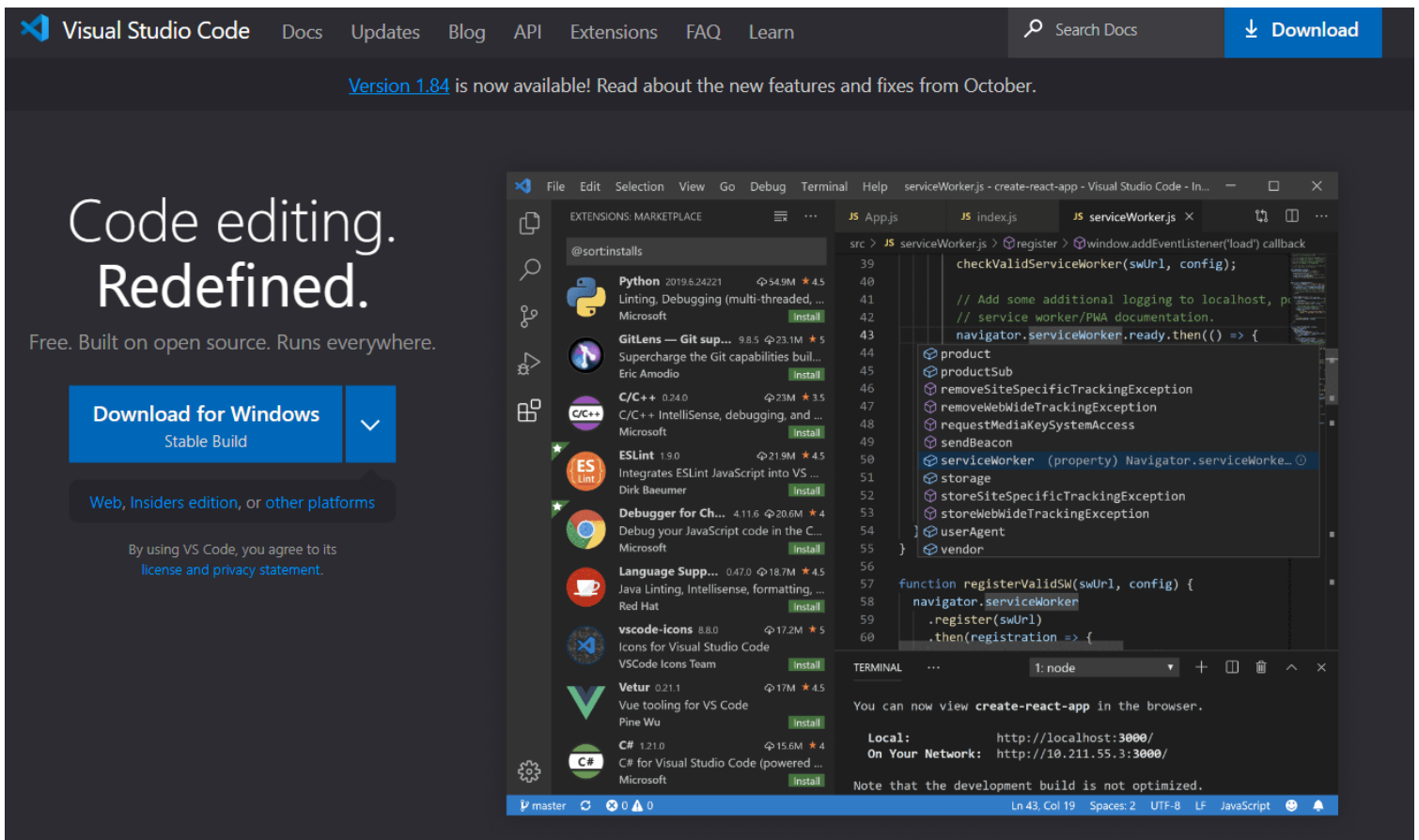
- [Install .NET SDK for Linux.](#)



- [Install VS Code for Linux.](#)

Once installed VS Code, you need to install a C# extension to facilitate us to code and debugging:

Install [C# extension](#).



1.1. Check the installed Fonts availability

Check that the directory with fonts `"/usr/share/fonts/truetype"` is exist. Also check that it contains `*.ttf` files.

If you don't see this folder, you may install "Microsoft TrueType core fonts" using terminal and command:

```
$ sudo apt install ttf-mscorefonts-installer
```

```
linuxconfig@linuxconfig-org: ~  
All done, no errors.  
Extracting cabinet: /var/lib/update-notifier/package-data-downloads/partial/verdan32.exe  
  extracting fontinst.exe  
  extracting fontinst.inf  
  extracting Verdanab.TTF  
  extracting Verdanai.TTF  
  extracting Verdanz.TTF  
  extracting Verdana.TTF  
  
All done, no errors.  
Extracting cabinet: /var/lib/update-notifier/package-data-downloads/partial/webdin32.exe  
  extracting fontinst.exe  
  extracting Webdings.TTF  
  extracting fontinst.inf  
  extracting Licen.TXT  
  
All done, no errors.  
All fonts downloaded and installed.  
Processing triggers for man-db (2.9.0-2) ...  
Processing triggers for fontconfig (2.13.1-2ubuntu2) ...  
linuxconfig@linuxconfig-org:~$
```

Read more about [TrueType Fonts and "How to install Microsoft fonts, How to update fonts cache files, How to confirm new fonts installation"](#) .

In next paragraphs we will explain in detail how to create simple console application. All of them are based on this VS Code guide:

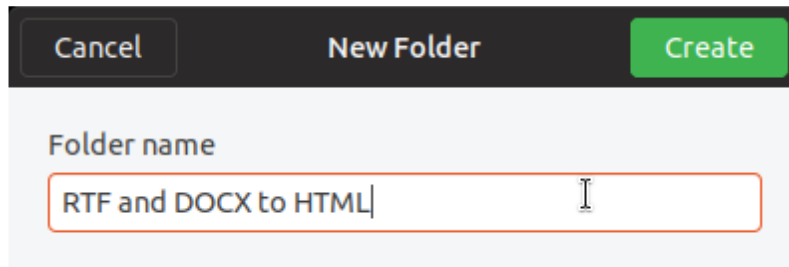
[Get Started with C# and Visual Studio Code](#)

Not only is possible to create .NET applications that will run on Linux using Linux as a developing platform. It is also possible to create it using a Windows machine and any modern Visual Studio version, as Microsoft Visual Studio Community 2022.

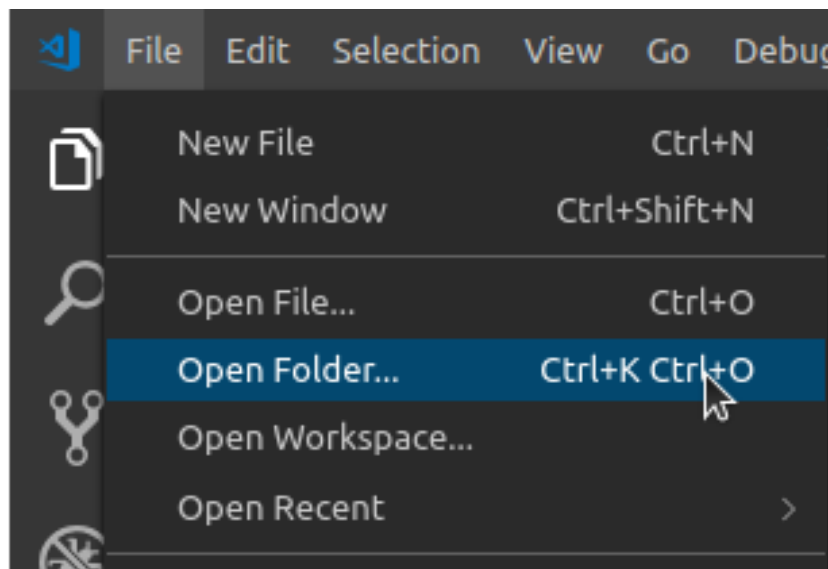
2. Creating “Convert RTF/DOCX to HTML” app

Create a new folder in your Linux machine with the name ***RTF and DOCX to HTML***.

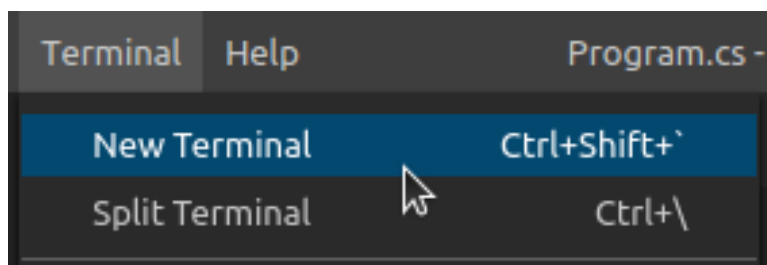
For example, let’s create the folder “***RTF and DOCX to HTML***” on Desktop (Right click-> New Folder):



Open VS Code and click in the menu ***File->Open Folder***. From the dialog, open the folder you’ve created previously:

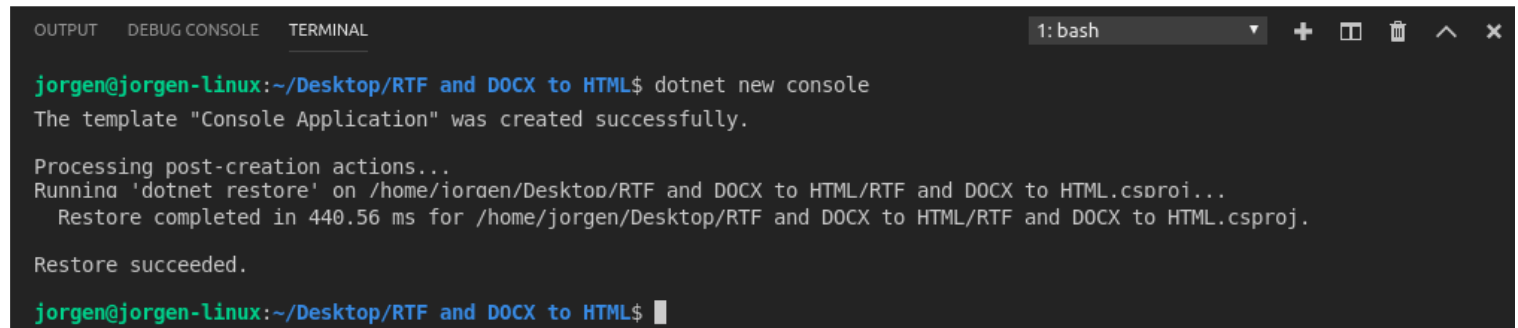


Now, open the integrated console – the Terminal: follow to the menu ***Terminal -> New Terminal*** (or press Ctrl+Shift+’):



Create a new console application, using **dotnet** command.

Type this command in the Terminal console: **dotnet new console**



```
OUTPUT  DEBUG CONSOLE  TERMINAL  1: bash
jorgen@jorgen-linux:~/Desktop/RTF and DOCX to HTML$ dotnet new console
The template "Console Application" was created successfully.

Processing post-creation actions...
Running 'dotnet restore' on /home/jorgen/Desktop/RTF and DOCX to HTML/RTF and DOCX to HTML.csproj...
Restore completed in 440.56 ms for /home/jorgen/Desktop/RTF and DOCX to HTML/RTF and DOCX to HTML.csproj.

Restore succeeded.

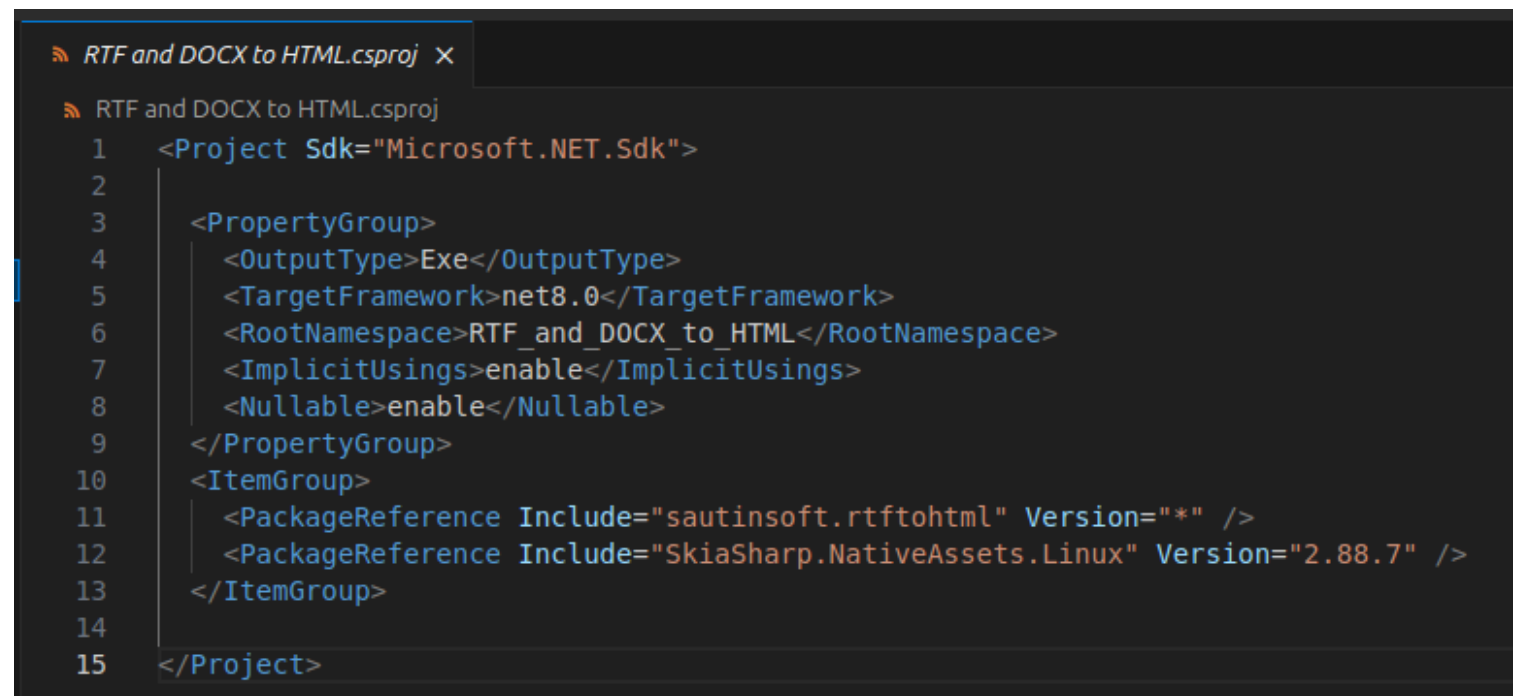
jorgen@jorgen-linux:~/Desktop/RTF and DOCX to HTML$
```

Now we are going to modify this simple application into an application that will convert rtf and docx files into HTML format.

First of all, we need to add the package reference to the **sautinsoft.rtftohtml** assembly using Nuget or the library SautinSoft.RtfToHtml.dll with additional references.

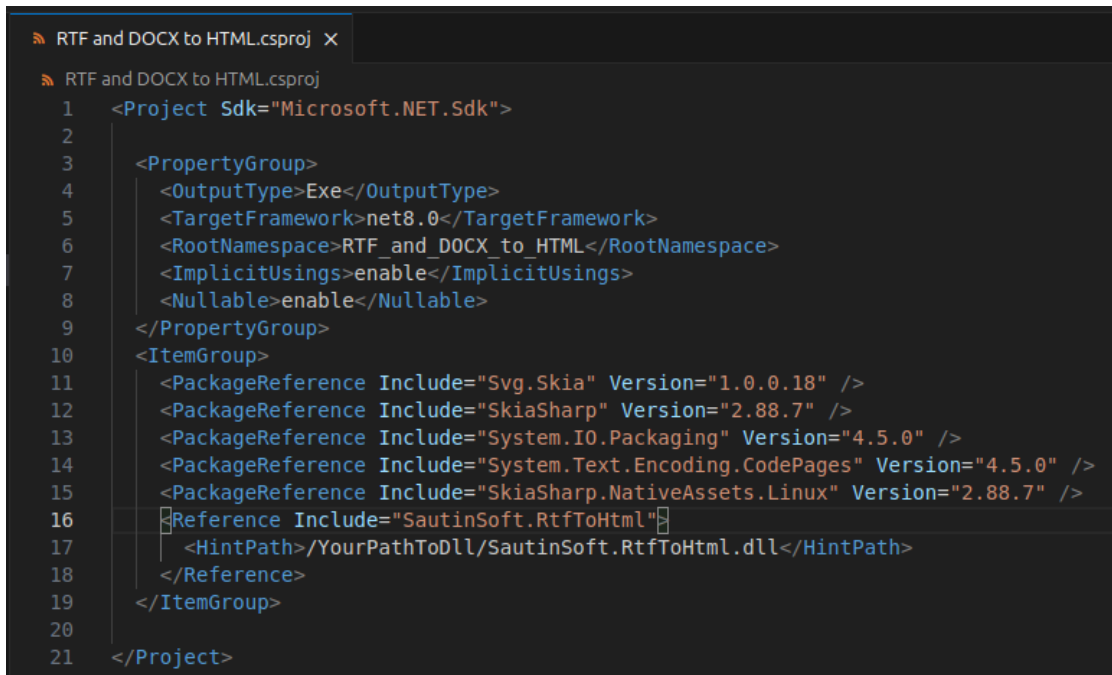
In order to do it, follow to the **Explorer** and open project file "**RTF and DOCX to HTML.csproj**" :

In the first case (NuGet):



```
RTF and DOCX to HTML.csproj x
RTF and DOCX to HTML.csproj
1  <Project Sdk="Microsoft.NET.Sdk">
2
3      <PropertyGroup>
4          <OutputType>Exe</OutputType>
5          <TargetFramework>net8.0</TargetFramework>
6          <RootNamespace>RTF_and_DOCX_to_HTML</RootNamespace>
7          <ImplicitUsings>enable</ImplicitUsings>
8          <Nullable>enable</Nullable>
9      </PropertyGroup>
10     <ItemGroup>
11         <PackageReference Include="sautinsoft.rtftohtml" Version="*" />
12         <PackageReference Include="SkiaSharp.NativeAssets.Linux" Version="2.88.7" />
13     </ItemGroup>
14
15 </Project>
```

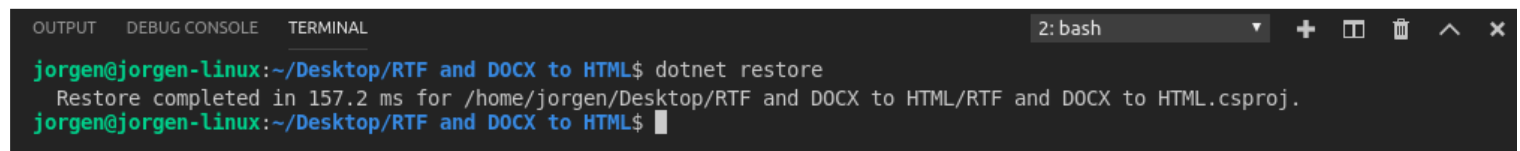
In the second case (SautinSoft.RtfToHtml.dll):



```
1 <Project Sdk="Microsoft.NET.Sdk">
2
3   <PropertyGroup>
4     <OutputType>Exe</OutputType>
5     <TargetFramework>net8.0</TargetFramework>
6     <RootNamespace>RTF_and_DOCX_to_HTML</RootNamespace>
7     <ImplicitUsings>enable</ImplicitUsings>
8     <Nullable>enable</Nullable>
9   </PropertyGroup>
10  <ItemGroup>
11    <PackageReference Include="Svg.Skia" Version="1.0.0.18" />
12    <PackageReference Include="SkiaSharp" Version="2.88.7" />
13    <PackageReference Include="System.IO.Packaging" Version="4.5.0" />
14    <PackageReference Include="System.Text.Encoding.CodePages" Version="4.5.0" />
15    <PackageReference Include="SkiaSharp.NativeAssets.Linux" Version="2.88.7" />
16    <Reference Include="SautinSoft.RtfToHtml" />
17    <HintPath>/YourPathToDll/SautinSoft.RtfToHtml.dll</HintPath>
18  </Reference>
19 </ItemGroup>
20
21 </Project>
```

At once as we've added the package references, we have to save the "**RTF and DOCX to HTML.csproj**" and restore the added packages.

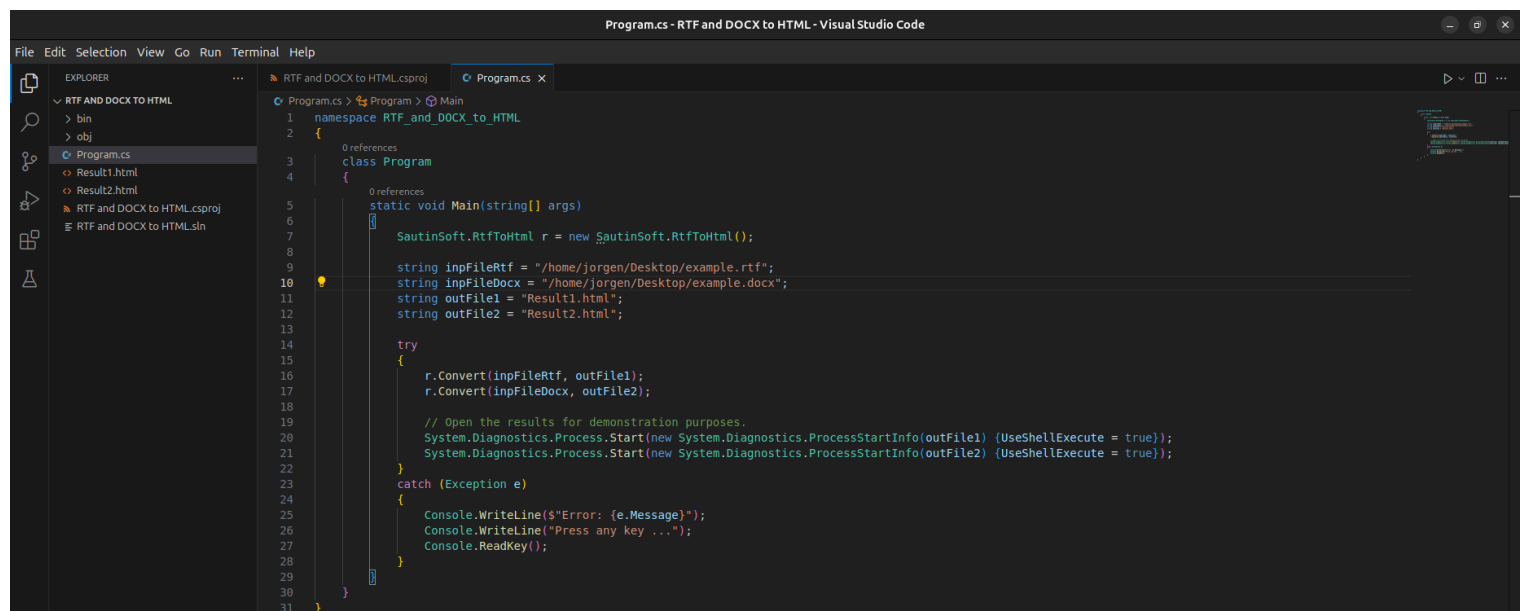
Follow to the **Terminal** and type the command: **dotnet restore**



```
OUTPUT  DEBUG CONSOLE  TERMINAL
jorgen@jorgen-linux:~/Desktop/RTF and DOCX to HTML$ dotnet restore
Restore completed in 157.2 ms for /home/jorgen/Desktop/RTF and DOCX to HTML/RTF and DOCX to HTML.csproj.
jorgen@jorgen-linux:~/Desktop/RTF and DOCX to HTML$
```

Good, now our application has all the references and we can write the code to convert DOCX and RTF documents into HTML format.

Follow to the **Explorer**, open the **Program.cs**, remove all the code and type the new:



```
1 namespace RTF_and_DOCX_to_HTML
2 {
3     0 references
4     class Program
5     {
6         0 references
7         static void Main(string[] args)
8         {
9             SautinSoft.RtfToHtml r = new SautinSoft.RtfToHtml();
10
11             string inFileRtf = "/home/jorgen/Desktop/example.rtf";
12             string inFileDocx = "/home/jorgen/Desktop/example.docx";
13             string outFile1 = "Result1.html";
14             string outFile2 = "Result2.html";
15
16             try
17             {
18                 r.Convert(inFileRtf, outFile1);
19                 r.Convert(inFileDocx, outFile2);
20
21                 // Open the results for demonstration purposes.
22                 System.Diagnostics.Process.Start(new System.Diagnostics.ProcessStartInfo(outFile1) {UseShellExecute = true});
23                 System.Diagnostics.Process.Start(new System.Diagnostics.ProcessStartInfo(outFile2) {UseShellExecute = true});
24             }
25             catch (Exception e)
26             {
27                 Console.WriteLine($"Error: {e.Message}");
28                 Console.WriteLine("Press any key ...");
29                 Console.ReadKey();
30             }
31 }
```

The code:

```
namespace RTF_and_DOCX_to_HTML
{
    class Program
    {
        static void Main(string[] args)
        {
            SautinSoft.RtfToHtml r = new SautinSoft.RtfToHtml();

            string inpFileRtf = "/home/jorgen/Desktop/example.rtf";
            string inpFileDocx = "/home/jorgen/Desktop/example.docx";
            string outFile1 = "Result1.html";
            string outFile2 = "Result2.html";

            try
            {
                r.Convert(inpFileRtf, outFile1);
                r.Convert(inpFileDocx, outFile2);

                // Open the results for demonstration purposes.
                System.Diagnostics.Process.Start(new
                System.Diagnostics.ProcessStartInfo(outFile1) {UseShellExecute = true});
                System.Diagnostics.Process.Start(new
                System.Diagnostics.ProcessStartInfo(outFile2) {UseShellExecute = true});
            }
            catch (Exception e)
            {
                Console.WriteLine($"Error: {e.Message}");
                Console.WriteLine("Press any key ...");
                Console.ReadKey();
            }
        }
    }
}
```

To make tests, we need the input RTF and DOCX documents. For our tests, let's place the files "example.rtf" and "example.docx" at the Desktop.



Launch our application to create a new HTML documents, type the command: ***dotnet run***

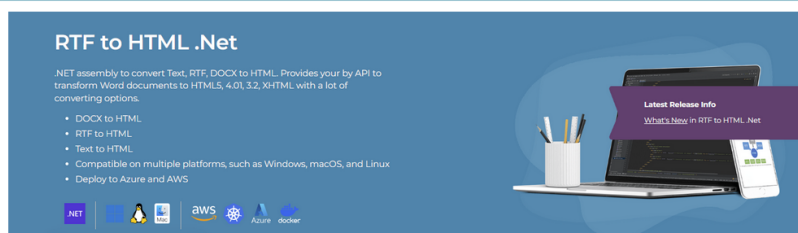
```
OUTPUT  DEBUG CONSOLE  TERMINAL
jorgen@jorgen-linux:~/Desktop/RTF and DOCX to HTML$ dotnet run
```

If you see the opening browser with the output documents, everything is fine and we can check the results produced by the [RTF to HTML .Net](#) library.



Welcome to SautinSoft!

This is «RTF to HTML .Net» sample



Subtitle

This rich text document content serves as basis for trying out various rich text formatting.

Header 1

Bold *italic* underlined ~~strikethrough~~ Nsubscript Nsuperscript

Sub header 1.1

Fonts:

Times new roman

Arial

Well done! You have created the “RTF/DOCX to HTML” application under Linux!

If you have any troubles or need extra code, or help, don’t hesitate to ask our SautinSoft Team at support@sautinsoft.com!