FREE eBook

LEARNING

asp.net-core-mvc

Free unaffiliated eBook created from
Stack Overflow contributors.

#asp.net-core-mvc

Table of Contents

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: asp-net-core-mvc

It is an unofficial and free asp.net-core-mvc ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official asp.net-core-mvc.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with asp.net-core-mvc

Remarks

This section provides an overview of what asp.net-core-mvc is, and why a developer might want to use it.

It should also mention any large subjects within asp.net-core-mvc, and link out to the related topics. Since the Documentation for asp.net-core-mvc is new, you may need to create initial versions of those related topics.

Examples

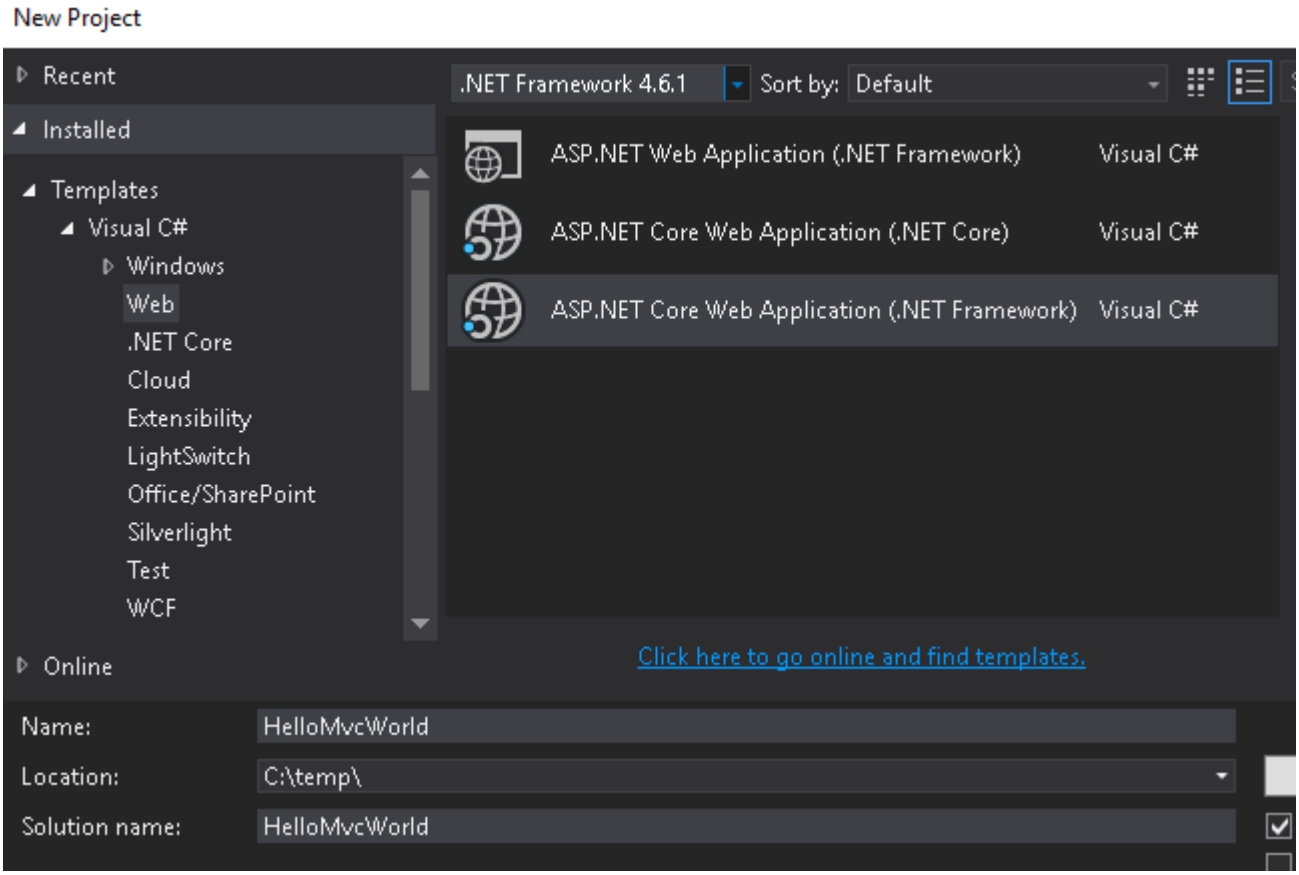
Installation or Setup

Installing Visual Studio

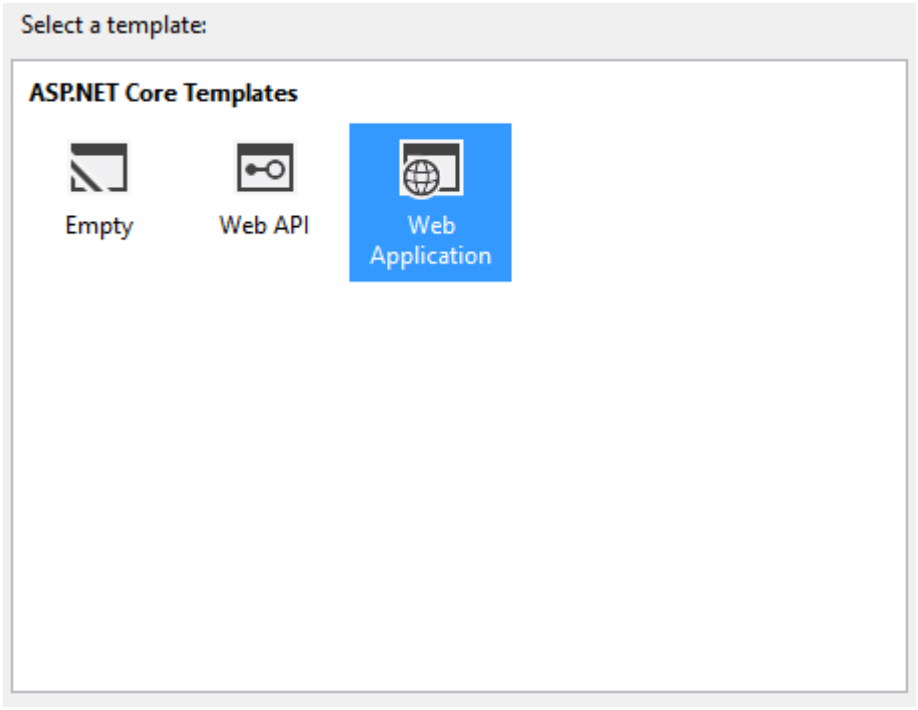
If you do not have Visual Studio installed, you can download the free Visual Studio Community Edition here. If you already have it installed, you can proceed to the next step.

Creating an ASP.NET Core MVC Application.

1. **Open Visual Studio.**
2. **Select File > New Project.**
3. **Select Web under the language of your choice** within the Templates section on the left.
4. **Choose a preferred Project type** within the dialog.
5. **Optional: Choose a .NET Framework you would like to target**
6. **Name your project** and indicate if you want to create a Solution for the project.
7. **Click OK** to create the project.



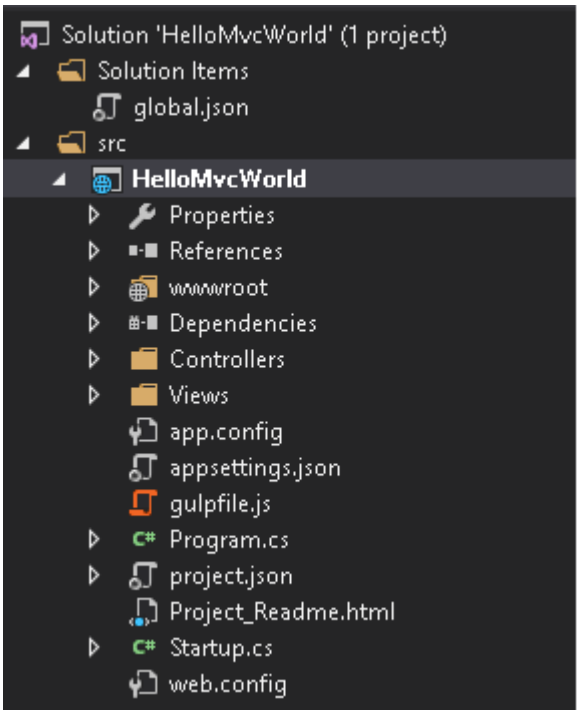
You will be presented with another dialog to select the template you want to use for the project :



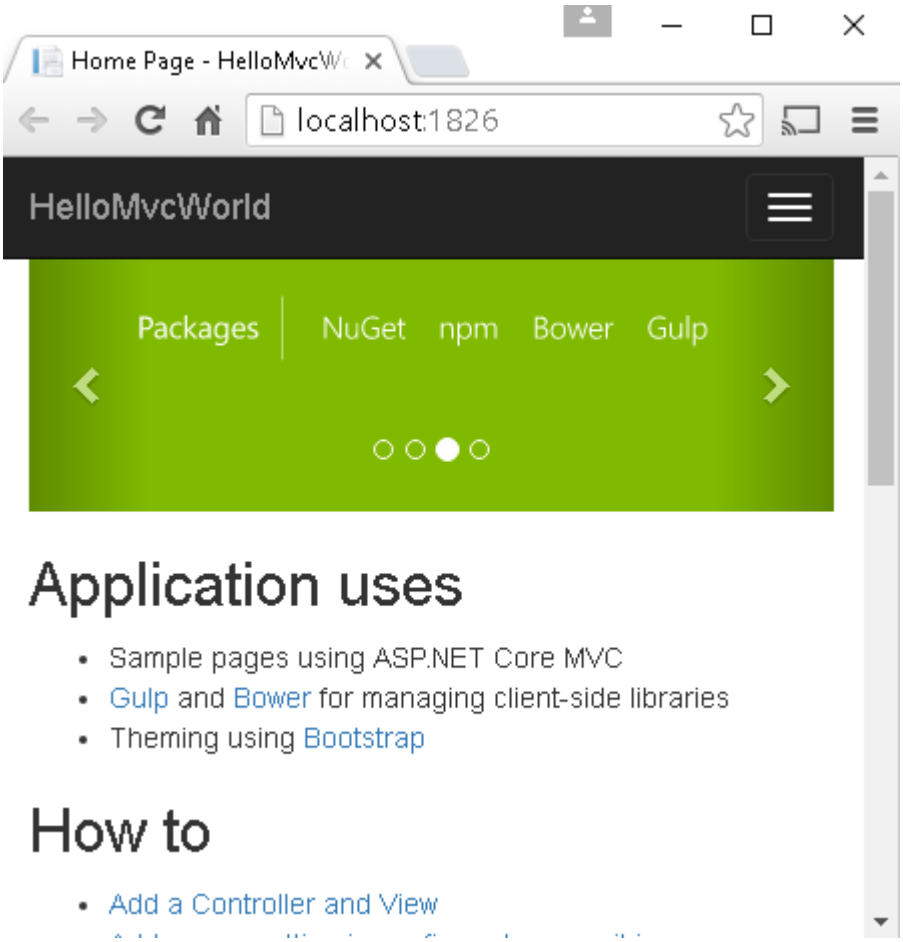
Each of the descriptions are self-explanatory. For this first project, **select Web Application**, which will contain all of the default configurations, authentication, and some existing content.

Since this is an introduction application and doesn't require any security or authentication, you can **change the authentication option to No Authentication** on the right-side of the dialog and **click OK to create the project**.

You should then see the new project within the Solution Explorer :



Press the F5 key to run the application and begin a debugging session, which will launch the application within your default browser :



You can now see that your project is up and running locally and is ready as a starting point for you to build your application.

PS: Used Getting started with asp.net-core topic from the asp.net-core Documentation.

Add MVC Middleware

If you created an empty project, or you still don't have mvc configured in your application, you can add dependency:

```
"Microsoft.AspNetCore.Mvc": "1.0.1"
```

To your `project.json` file under `"dependencies"`.

And register MVC middleware in your Startup class:

```
public void ConfigureServices(IServiceCollection services)
{
    ...
    services.AddMvc();
}
```

Note that we have both `services.AddMvc()` and `services.AddMvcCore()`. If you are starting with `asp.net core`, or you want it the way it is, you should keep with `services.AddMvc()`. But if you want an advanced experience, you can start with a minimal MVC pipeline and add features to get a customized framework using `services.AddMvcCore()`. See this discussion for more information about `AddMvcCore`

```
public void ConfigureServices(IServiceCollection services)
{
    services
        .AddMvcCore()
        .AddAuthorization()
        .AddJsonFormatters(j => j.Formatting = Formatting.Indented);
}
```

Now you can tell your application builder to use the mvc:

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env, ILoggerFactory
loggerFactory)
{
    ...
    app.UseMvc();
}
```

or with default routing:

```
app.UseMvc(routes =>
{
    routes.MapRoute(
        name: "default",
        template: "{controller=Home}/{action=Index}/{id?}");
});
```

Dependency injection basics

Almost any controller needs some external dependencies to work. Here is a way to configure a dependency object (or its factory) and pass it to a controller. Doing so will help to sustain a separation of concerns, keep code clear and testable.

Say, we have an interface and its implementation that needs some values from config in its constructor:

```
public interface ISomeDependency
{
    async Task<IEnumerable<string>> GetItemsAsync(string key);
}

public class SomeDependency : ISomeDependency
{
    public SomeDependency(string connectionString)
    {
        ...
    }
    ...
}
```

It's used in some controller class:

```
public class SomeController : Controller
{
    private reanonly ISomeDependency dependency;

    public SomeController(ISomeDependency dependency)
    {
        ...
        this.dependency = dependency;
    }

    ...

    public async Task<IEnumerable<string>> Get(string key) =>
        await dependency.GetItemsAsync(key);
}
```

One can inject this dependency in the controller constructor calling `services.AddTransient` inside `Startup.ConfigureServices` method:

```
public class Startup
{
    public Startup(IHostingEnvironment env)
    {
        var builder = new ConfigurationBuilder().
            .SetBasePath(env.ContentRootPath)
            .AddJsonFile("appsettings.json", optional: true, reloadOnChange: true)
        ...
        Configuration = builder.Build();
    }

    public IConfigurationRoot Configuration { get; }

    public void ConfigureServices(IServiceCollection services)
```

```
    {
        ...
        services.AddTransient(serviceProvider =>
            new MyDependency(Configuration["Data:ConnectionString"]));
    }

    ...
}
```

Here `Data:ConnectionString` is a path to a setting in `appsettings.json` file:

```
{
  ...
  },
  "Data": {
    "ConnectionString": "some connection string"
  }
}
```

Lifetime management

To manage a lifetime of the injected object, along with `AddTransient` another two options exist: `AddSingleton` and `AddScoped`. The last one means that lifetime of the object is scoped to a HTTP request.

Versions

Official roadmap @ Github

Version	Announcements	Release Date
RC1*	1.0.0-rc1	2015-11-01
RC2*	1.0.0-rc2	2016-05-16
1.0.0	1.0.0	2016-06-27
1.0.1	1.0.1	2016-09-13
1.0.1	1.0.1	2016-09-13
1.1	1.1.0	Q4 2016 / Q1 2017
1.2	1.2.0	Q1 2017 / Q2 2017

** References to yearly quarters (Q1, Q2, Q3, Q4) are calendar-based*

Read Getting started with asp.net-core-mvc online: https://riptutorial.com/asp-net-core-mvc/topic/2174/getting-started-with-asp-net-core-mvc

Chapter 2: Change default view location

Introduction

In ASP.NET MVC, the views are placed by default in the `Views` folder. Sometimes you want to change this locations and store the views somewhere else.

Examples

Create a View Location Expander

To be able to change the view location, you need to implement the `IViewLocationExpander`. The `ExpandViewLocations` method returns an `IEnumerable<string>` containing the different locations where to search, with

```
public class MyViewLocationExpander : IViewLocationExpander
{
    public IEnumerable<string> ExpandViewLocations(ViewLocationExpanderContext context,
IEnumerable<string> viewLocations)
    {
        yield return "/CustomViewFolder/{1}/{0}.cshtml";
        yield return "/SharedFolder/{0}.cshtml";
    }

    public void PopulateValues(ViewLocationExpanderContext context)
    {
    }
}
```

Register the View Location Expander

You now need to register the Expander, in order for it to be used by the Razor View Engine. Just add this in the `ConfigureServices` of your `Startup` class.

```
public void ConfigureServices(IServiceCollection services)
{
    services.Configure<RazorViewEngineOptions>(options => {
        options.ViewLocationExpanders.Add(new MyViewLocationExpander());
    });
}
```

Read Change default view location online: https://riptutorial.com/asp-net-core-mvc/topic/8669/change-default-view-location

Chapter 3: Setup and install .Net Core MVC with Visual studio code and quick start .net core mvc hello world.

Introduction

This article give idea's about setup and installing Asp.Net core with visual studio code. Also create basic MVC template and debugging.

Steps involved below...

Step 1 - installing Visual studio code.

Step 2 - Configuring .Net core and C#.

Step 3 - Create Basic MVC Template.

Step 4 - Execute and Debug the application.

Remarks

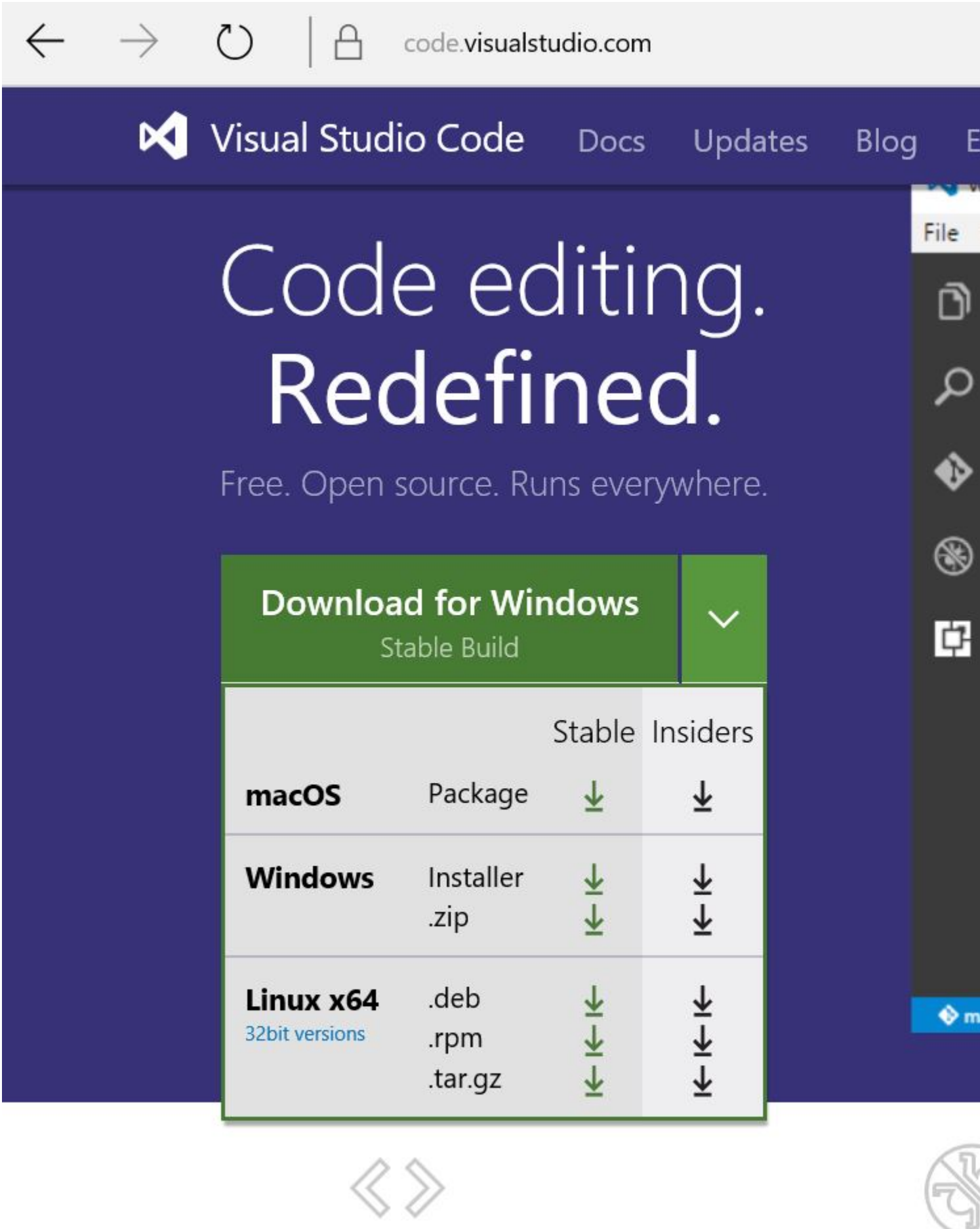
This article is about to setup from scratch with visual studio code open source and create and debug basic .net core mvc applications.

- File location used above is change as per users, No constraint.
- Need internet for downloading setups.

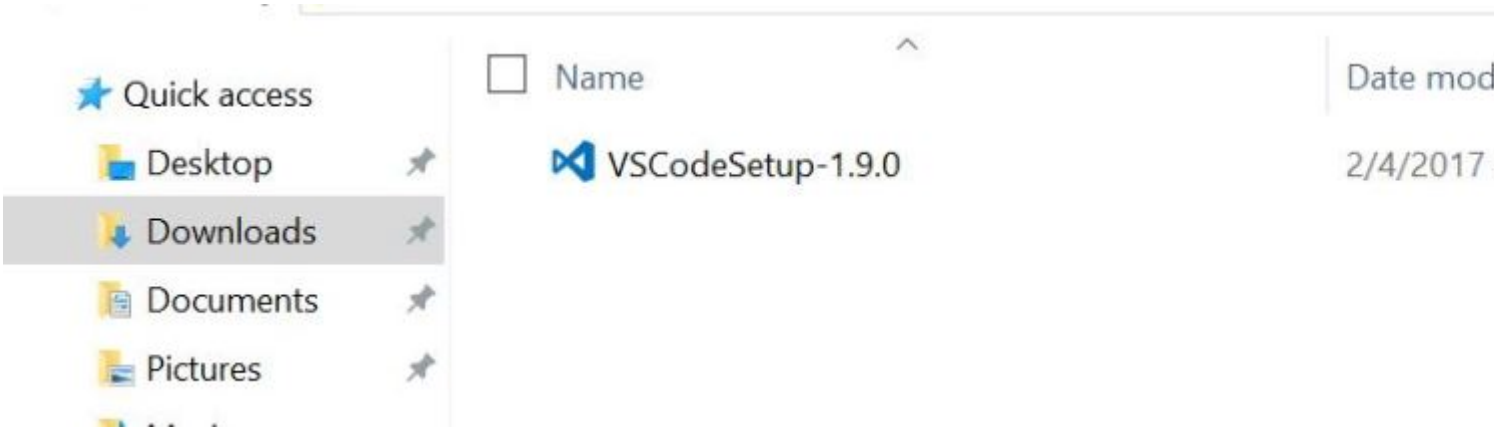
Examples

Step 1 - Visual studio code installation

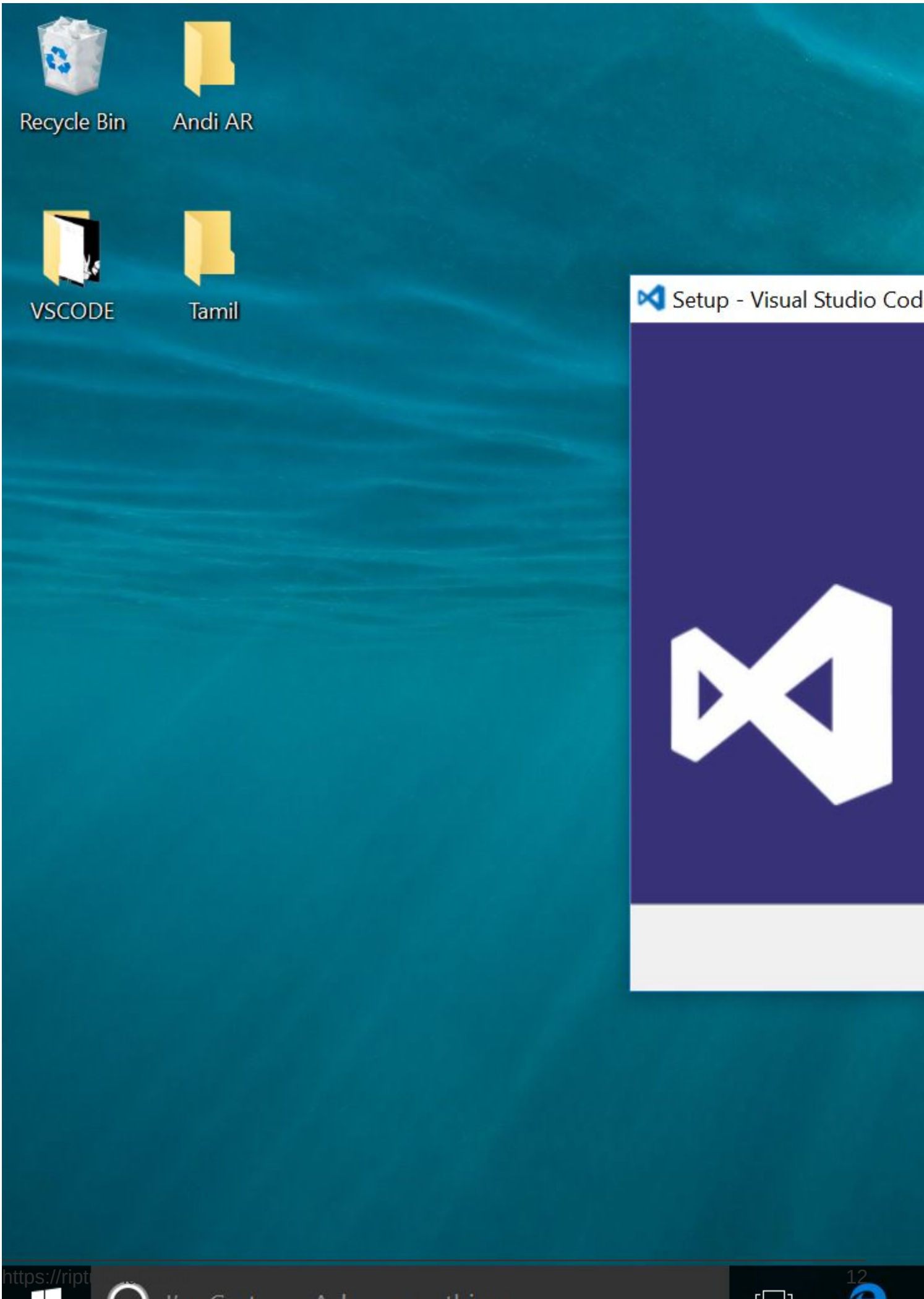
- Download visual studio code from here Visual studio code. Select your target installer[mac|windows|linux].


code.visualstudio.com
Visual Studio Code
Docs
Updates
Blog
Code editing.
Redefined.
Free. Open source. Runs everywhere.
Download for Windows
Stable Build
Stable
Insiders
macOS
Package
Windows
Installer
.zip
Linux x64
32bit versions
.deb
.rpm
.tar.gz
File

- Go to downloaded file in your local.

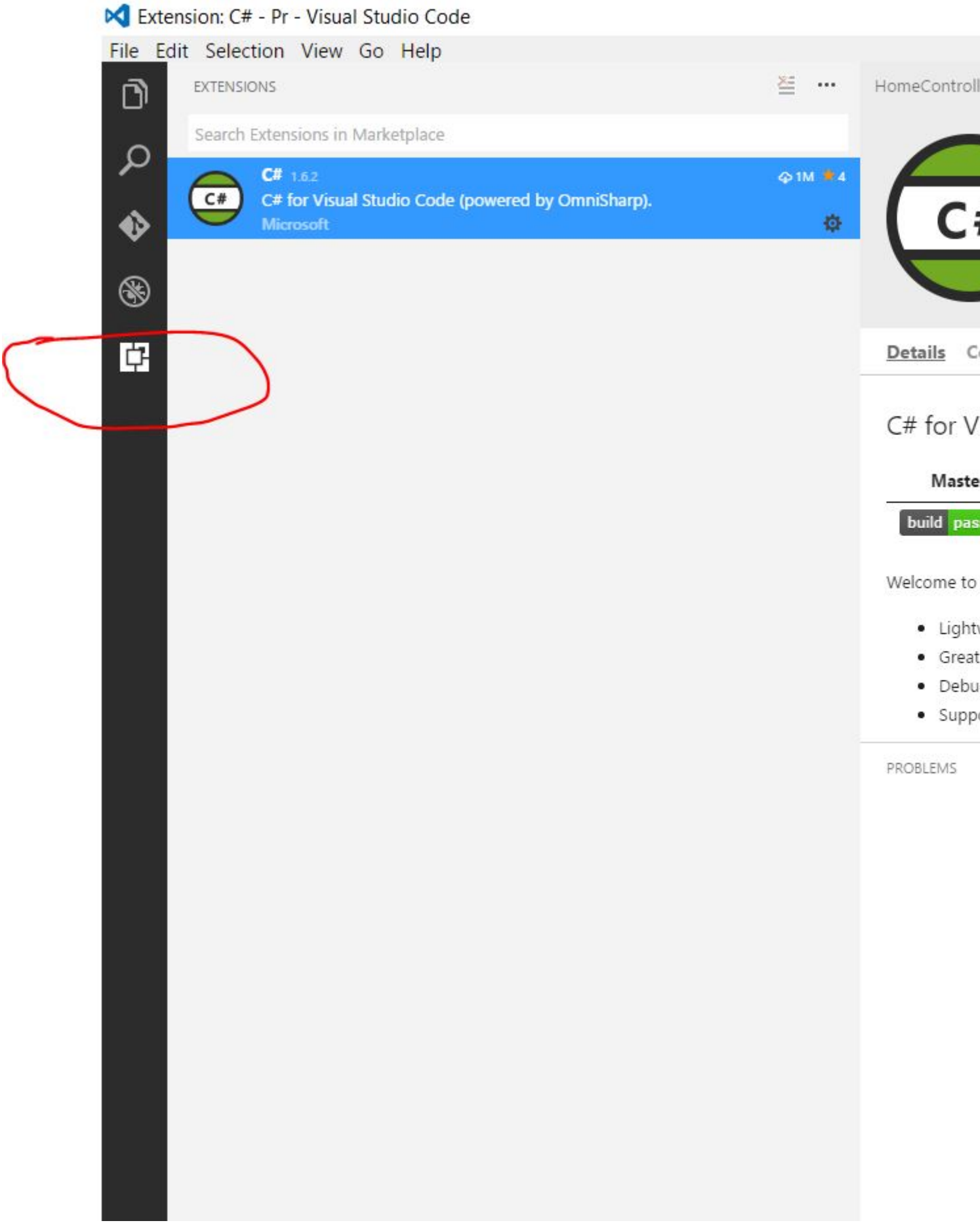


- Below steps in volved for installing


Recycle Bin
Andi AR
VSCODE
Tamil
Setup - Visual Studio Cod
I'm Cortana. Ask me anything.

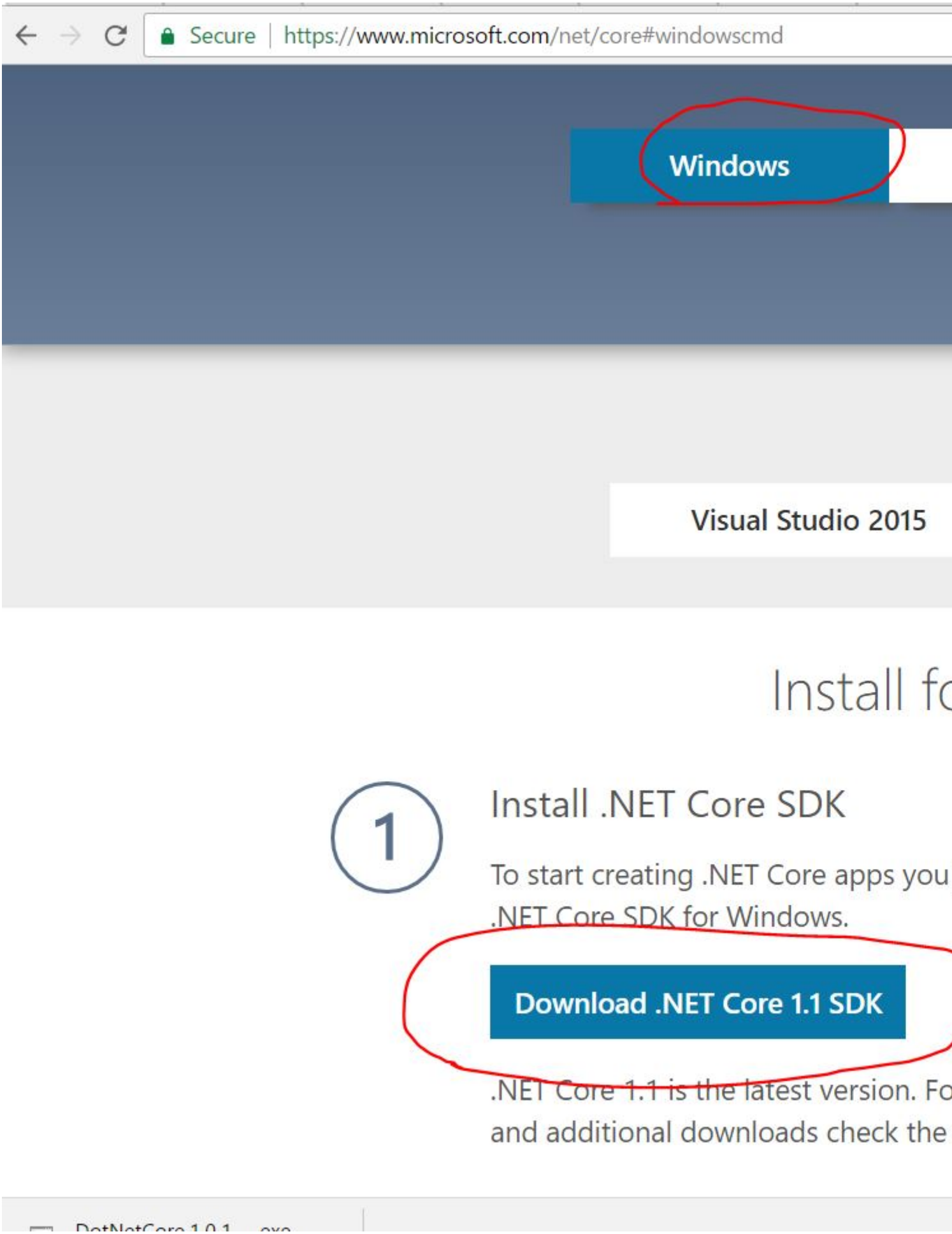
2. Press [**ctrl + P**]
3. paste "**ext install csharp**" this and hit.

Once done above steps , C# extension available in VS Code.

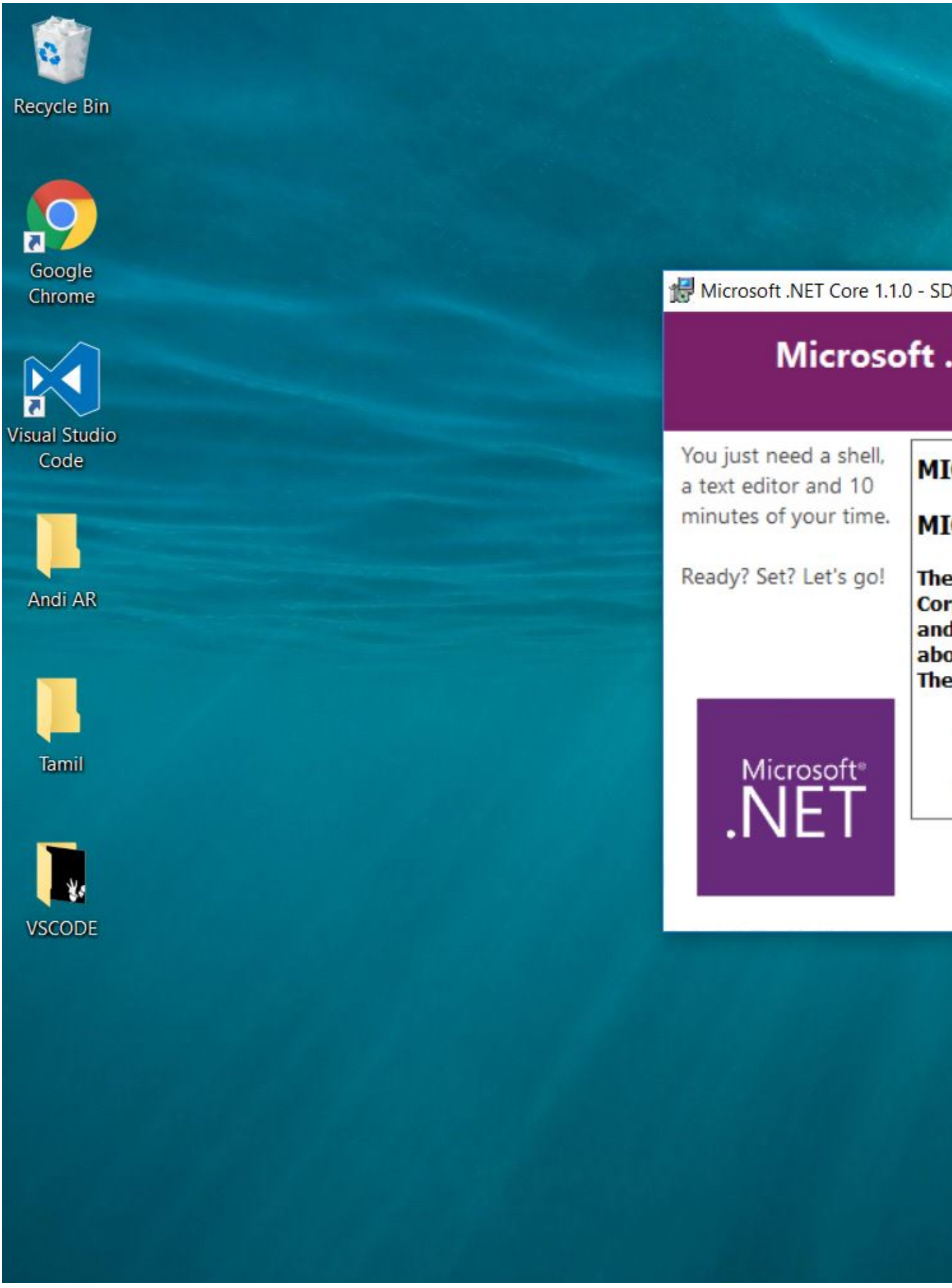


- Now configure .net core.

Download .net core sdk from here. Choose Windows=>CommandLine.


Secure | https://www.microsoft.com/net/core#windowscmd
Windows
Visual Studio 2015
Install fo
1
Install .NET Core SDK
To start creating .NET Core apps you
.NET Core SDK for Windows.
Download .NET Core 1.1 SDK
.NET Core 1.1 is the latest version. Fo
and additional downloads check the

Install the sdk like below.


Recycle Bin
Google Chrome
Visual Studio Code
Andi AR
Tamil
VSCODE
Microsoft .NET Core 1.1.0 - SD
Microsoft .
You just need a shell, a text editor and 10 minutes of your time.
Ready? Set? Let's go!
Microsoft
.NET

Credits

S. No	Chapters	Contributors
1	Getting started with asp.net-core-mvc	Community, Ole K, Rafael Marques, Set, stop-cran, tmg, Zach Becknell
2	Change default view location	glacasa
3	Setup and install .Net Core MVC with Visual studio code and quick start .net core mvc hello world.	Andi AR