**Dependency Injection(DI) Using Structure Map IOC Container in .NET Core**

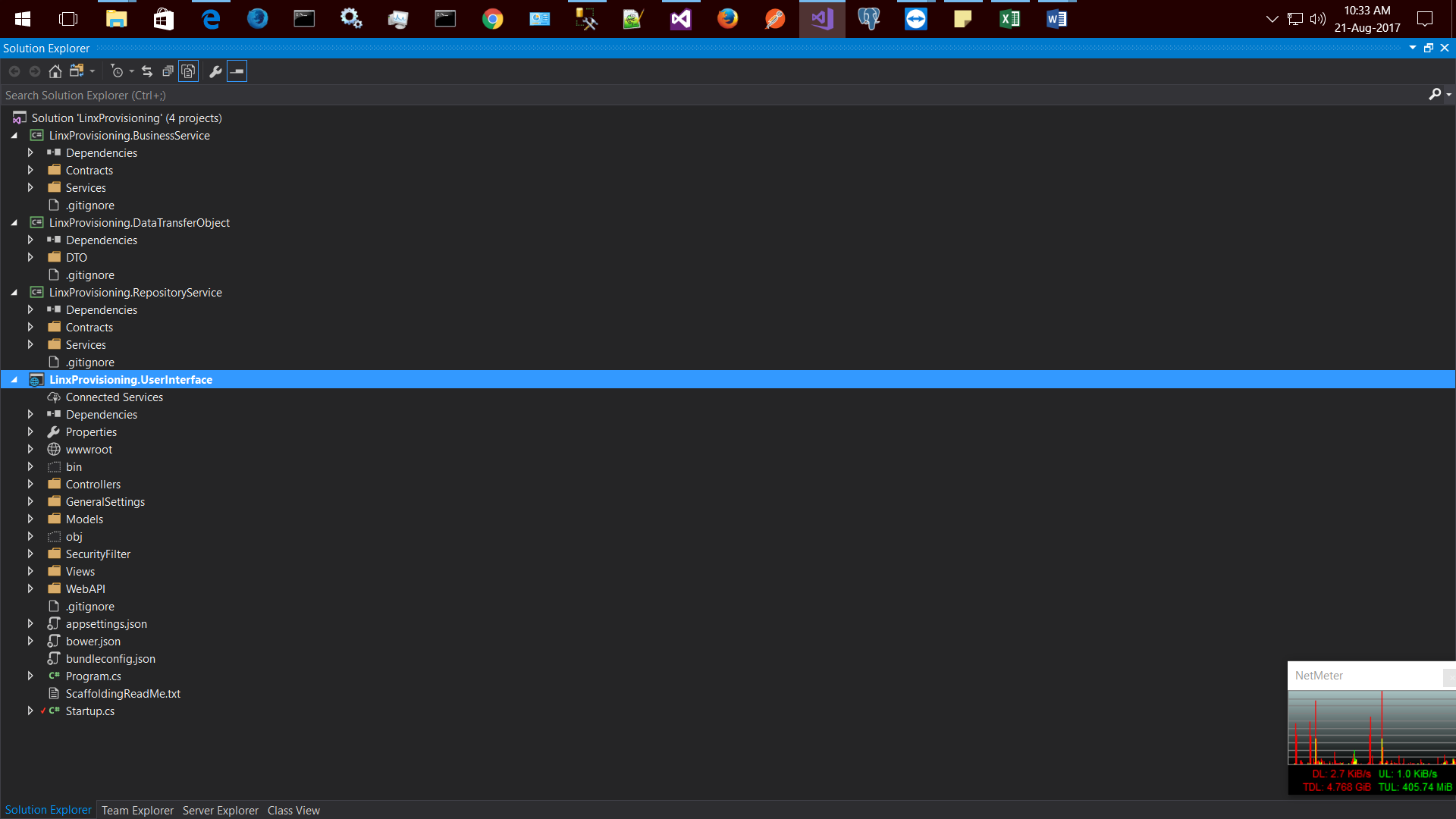
IMPORTANT THINGS TO NOTE;

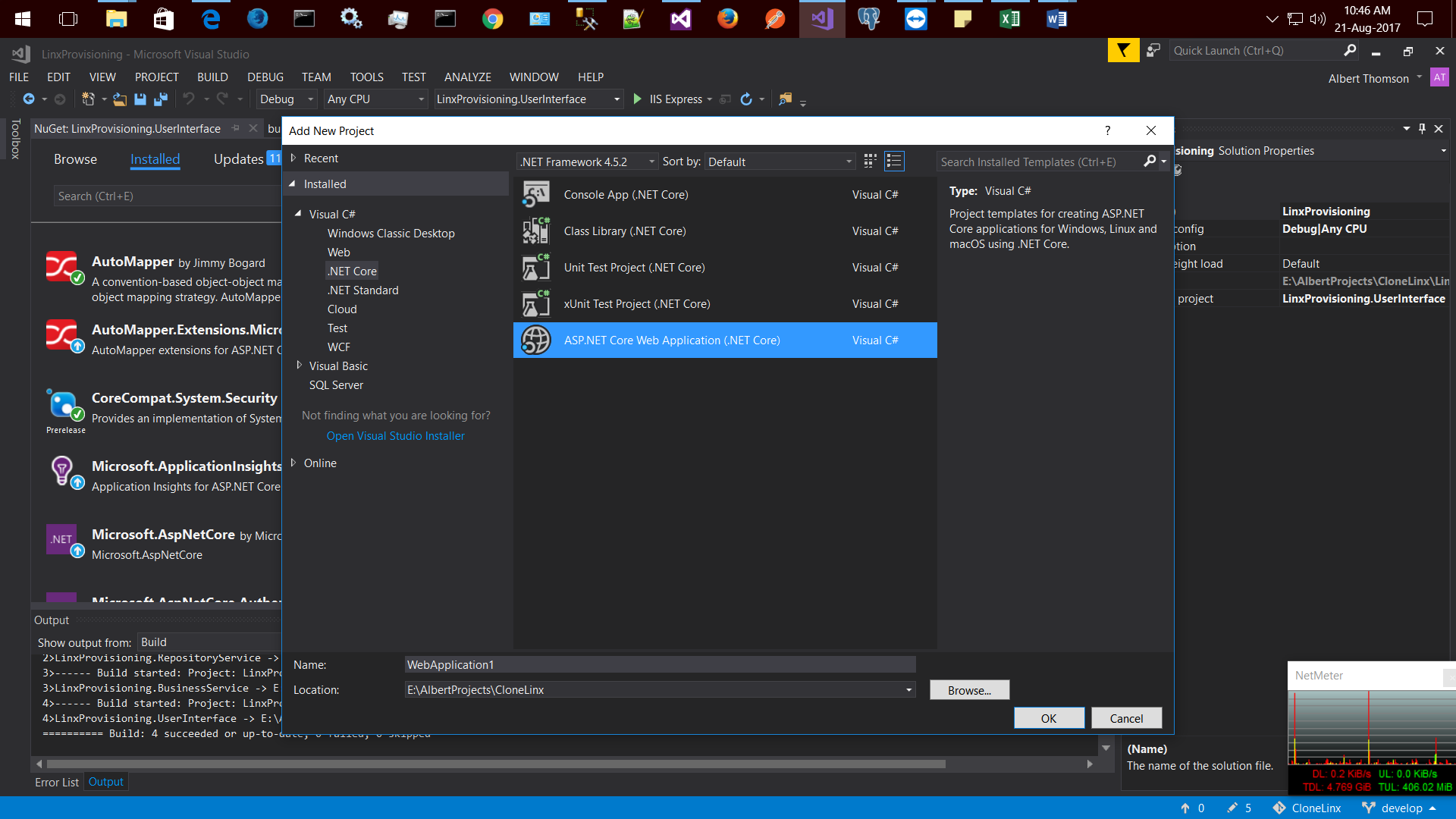
Structure map is another IOC container just like unity.

Unity does not support .NET Core so we use Structure Map.

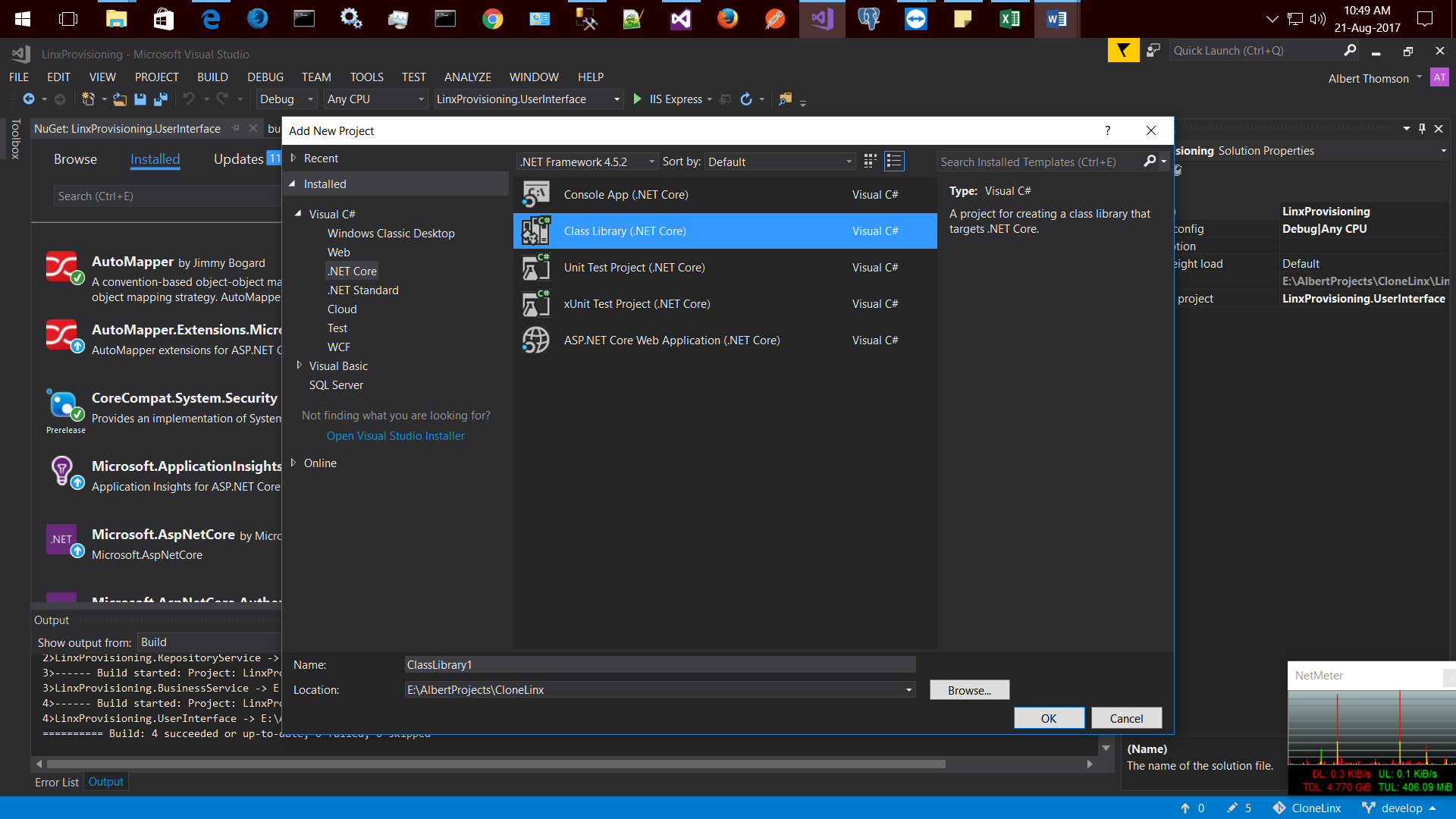
Before proceeding with DI in .net core project please refer the document “DI using unity “.

Here is a solution with 4 .net core projects

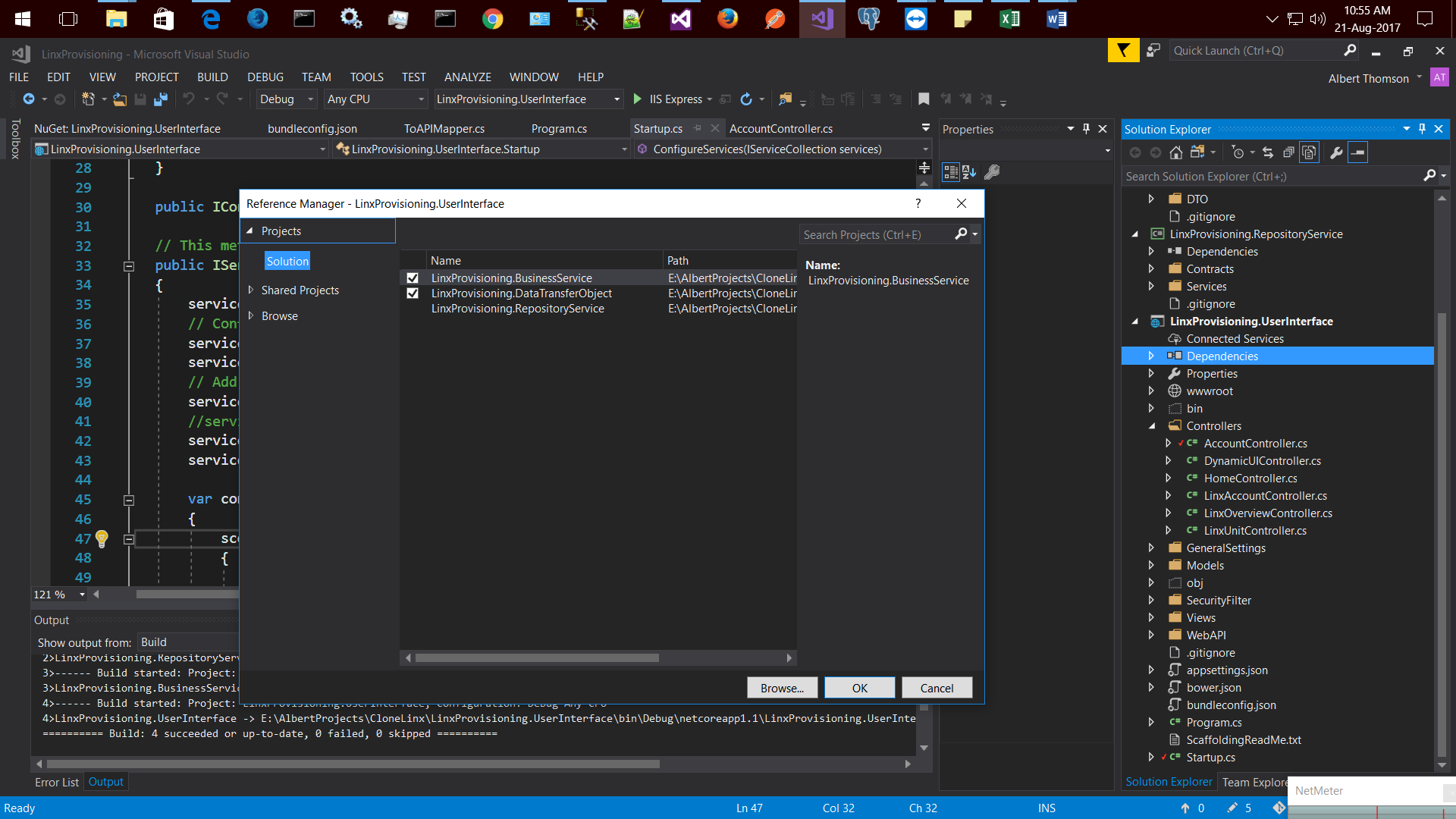


For User Interface project please choose ASP.NET Core Web Application

For other projects (Business, repository and DTO) please choose Class Library (.net core)



In order to refer a project assembly, right click on Dependencies and choose required libraries as follows



After linking projects,we need to install required packages from nuget.

Please note that with .net core project they have shipped another package manager called Bower.

Use nuget for server side dll packages and bower for client side packages(jquery,bootstrap etc).

STEP 1.Installl StructureMap.Microsfot.DependencyInjection from nuget.

STEP 2.Every .net core project starts with program.cs class execution,

In this it calls for startup class which have all the configuring methods,

Find out the method ConfigureServices and change its return type from void to IServiceProvider

This is the place where we register all service that we need.

The structure map registration code as follows

var container = new Container(scope =>

{

//scan for loaded assemblies

scope.Scan(x =>

{

x.Assembly("LinxProvisioning.BusinessService");

x.WithDefaultConventions();

});

scope.Scan(x =>

{

x.Assembly("LinxProvisioning.RepositoryService");

x.WithDefaultConventions();

});

});

container.Populate(services);

since we have changed its return type to IServiceProvider so return

return container.GetInstance<IServiceProvider>();

STEP3.now you can use constructor injection in both webapps and in web apis.