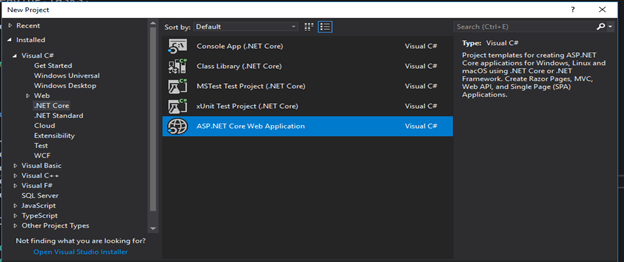
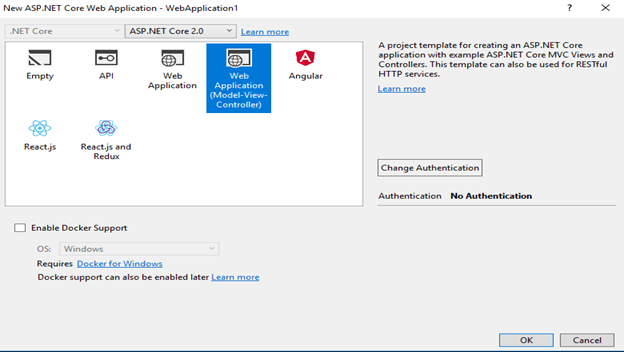
**Step 1 - Creating an ASP.NET Core MVC project**.

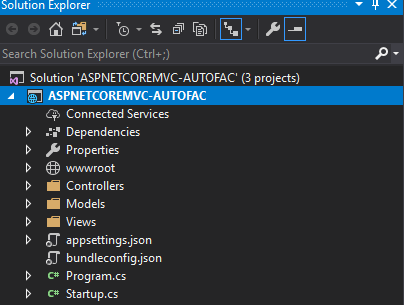
Open VS 2017 go to File->New->Project where you have to select ASP.NET core web application template and give any name you want, here I’ve given ASPNETCOREMVC-AUTOFAC.



And click ok it will display the Web application template where you have to select web application(MVC) then click ok.

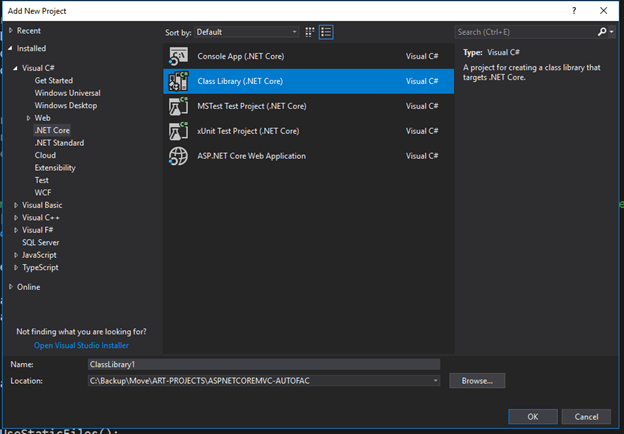


Now solution ready like below,

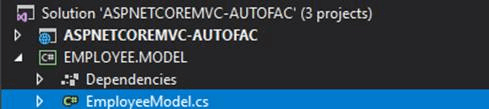


**Step 2 - Creating a Separate Project called Employee.Model**

Just right click on your solution and choose to add New Project option you’ll get below the screen where you have to select classLibrary(.NETCore) & given name EMPLOYEE.MODEL and click ok.



After you create you will get to see this project added under solution. Now try to add EmployeeModel class file under this project. Once you create a class file in solution will see like below,



Inside EmployeeModel.cs file paste below properties,

1. using System;
2. namespace EMPLOYEE.MODEL {
3. publicclass EmployeeModel {
4. publicint EmpId {
5. get;
6. set;
7. }
8. **public** string Name {
9. get;
10. set;
11. }
12. publicint Salary {
13. get;
14. set;
15. }
16. **public** string Address {
17. get;
18. set;
19. }
20. }
21. }

**Step 3 - Create Service Employee.Service**

So far we have created a web controller and model, now we have to create another project called Employee.Service. which is responsible to interact with the database. Do the same steps what we did in step 2 for creating a new project. But here we have to add two files.

* IEmployee.cs interface file
* Employee.cs file

These files are responsible to achieve actual crud operation.

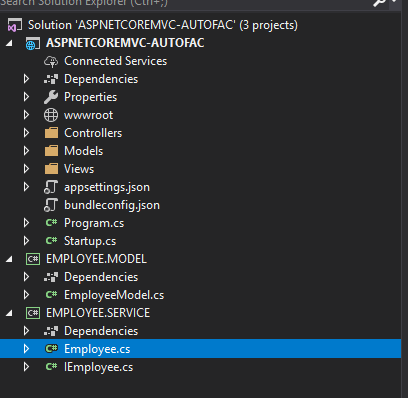
**IEmployee.cs**

1. using System;
2. using System.Collections.Generic;
3. using System.Text;
4. using EMPLOYEE.MODEL;
5. namespace EMPLOYEE.SERVICE {
6. publicinterface IEmployee {
7. EmployeeModel GetEmployee();
8. **int** PostEmployee(EmployeeModel employee);
9. **int** UpdateEmployee(EmployeeModel employee);
10. **int** DeleteEmployee(**int** Id);
11. }
12. }

**Employee.cs**

1. using EMPLOYEE.MODEL;
2. using System;
3. namespace EMPLOYEE.SERVICE {
4. publicclass Employee: IEmployee {
5. **public** EmployeeModel GetEmployee() {
6. returnnew EmployeeModel();
7. }
8. publicint PostEmployee(EmployeeModel employeeModel) {
9. //Db stuff goes here
10. **return**
11. }
12. publicint UpdateEmployee(EmployeeModel employeeModel) {
13. //Db stuff goes here
14. **return** 1;
15. }
16. publicint DeleteEmployee(**int** Id) {
17. //Db stuff goes here
18. **return** 1;
19. }
20. }
21. }

So once you set up three projects under solution it would look like below,



**Step 4 - Add Autofac NuGet library to a web project**

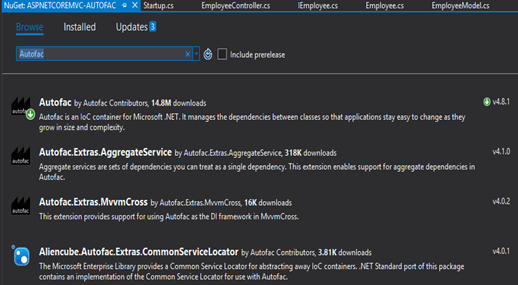
Go to your web project to add Autofac NuGet packages here our web project name is

**ASPNETCOREMVC-AUTOFAC**

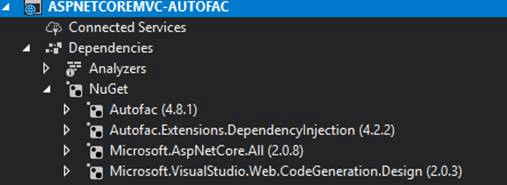
Right click on this project and choose *ManageNugetPackages*option it will show below a screen

Here you have to search below two packages

* Autofac
* Extension.DependencyInjection



Once you add these two packages it will add under the NuGet folder of your project like below



Now you have to tweak in starup.cs file. Go to Starup.cs file paste below snippet under ConfigureService Method

1. **public** IServiceProvider ConfigureServices(IServiceCollection services) {
2. services.AddMvc();
3. // Create the container builder to register services with Autofac container.
4. **var** builder = **new** ContainerBuilder();
5. builder.Populate(services);
6. builder.RegisterType < Employee > ().As < IEmployee > ().InstancePerDependency();
7. **this**.ApplicationContainer = builder.Build();
8. // Create the IServiceProvider based on the container.
9. returnnew AutofacServiceProvider(**this**.ApplicationContainer);
10. }

The above ConfigureServices() returns IServiceProvider and uses Autofac container for resolving dependencies.The builder.Populate(services) will copy all the existing dependencies added into the IServiceCollection into Autofac container. That’s it.Asp.net core will use Autofac to resolve dependencies.

Now go to EmployeeController under web project and inject IEmployee interface as shown below

1. using System;
2. using System.Collections.Generic;
3. using System.Linq;
4. using System.Threading.Tasks;
5. using EMPLOYEE.MODEL;
6. using EMPLOYEE.SERVICE;
7. using Microsoft.AspNetCore.Mvc;
8. namespace ASPNETCOREMVC\_AUTOFAC.Controllers {
9. publicclass EmployeeController: Controller {
10. **private** IEmployee \_employee;
11. **public** EmployeeController(IEmployee employee) {
12. \_employee = employee;
13. }
14. **public** EmployeeModel GetEmployee() {
15. **return** \_employee.GetEmployee();
16. }
17. **public** EmployeeModel PostEmployee(EmployeeModel employeeModel) {
18. **return** \_employee.GetEmployee();
19. }
20. **public** EmployeeModel UpdateEmployee(EmployeeModel employeeModel) {
21. **return** \_employee.GetEmployee();
22. }
23. publicint DeleteEmployee(**int** Id) {
24. **return** Id;
25. }
26. }
27. }

In the constructor, we injected the IEmployee interface.so Employee controller depends on IEmployee interface it would resolve by Autofac container with help of configuration we did in Starup.cs file.