

SYED AWASE KHIRNI

Example 17:clean architecture using cqrs/mediatr – Wiring them up!

Learning Outcomes:

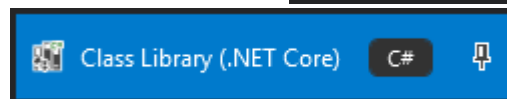
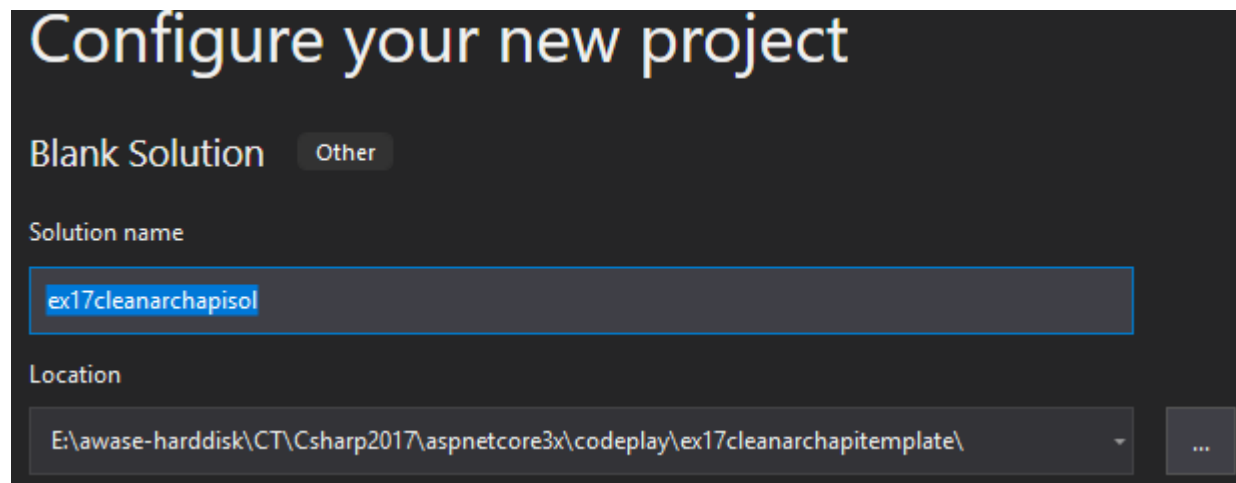
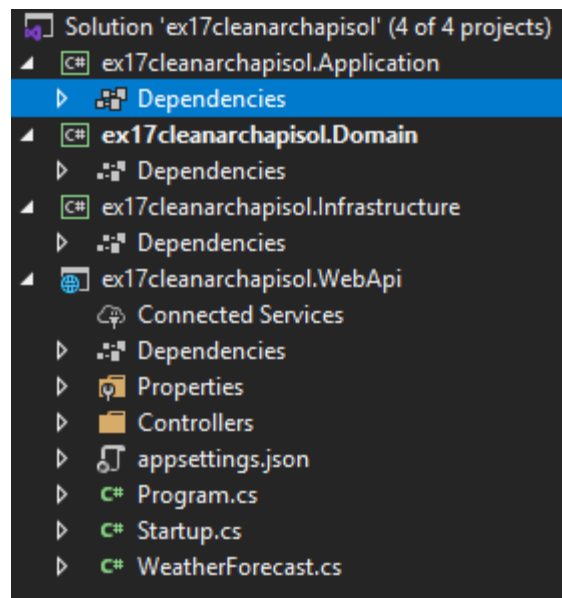
1. Creating a Clean Architecture/Onion architecture template MVC template using mediatr + CQRS design pattern.

Clean Architectural Requirements

- Independent of any framework
- Decoupled from any third party frameworks
- Pluggable WebUI/ WebUI independence – easy to switch between MVC/API/Angular/Vuejs/Reactjs
- Database Independence- easy portability to any database
- Domain driven design
- Separation of concerns – applying CQRS design pattern – segregating commands and queries for improved performance and scalability.
- Unit testing, functional testing and integration testing

Essential Libraries

- Entity Framework Core 3.x
- AutoMapper
- Dependency Injection
- Logging using Serilog/Nlog/StackExchange.Exceptional
- CQRS with MediatR
- Fluent Validations
- MVC using Razor web pages
- Swagger/Open API
- Error Handling
- Unit Testing with Nunit
- Integration testing with NUnit

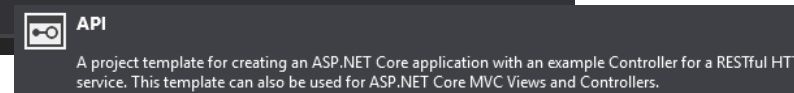
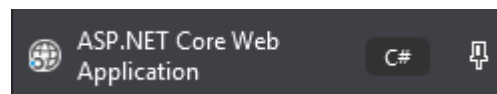


ex17cleanarchapisol.Domain

ex17cleanarchapisol.Application

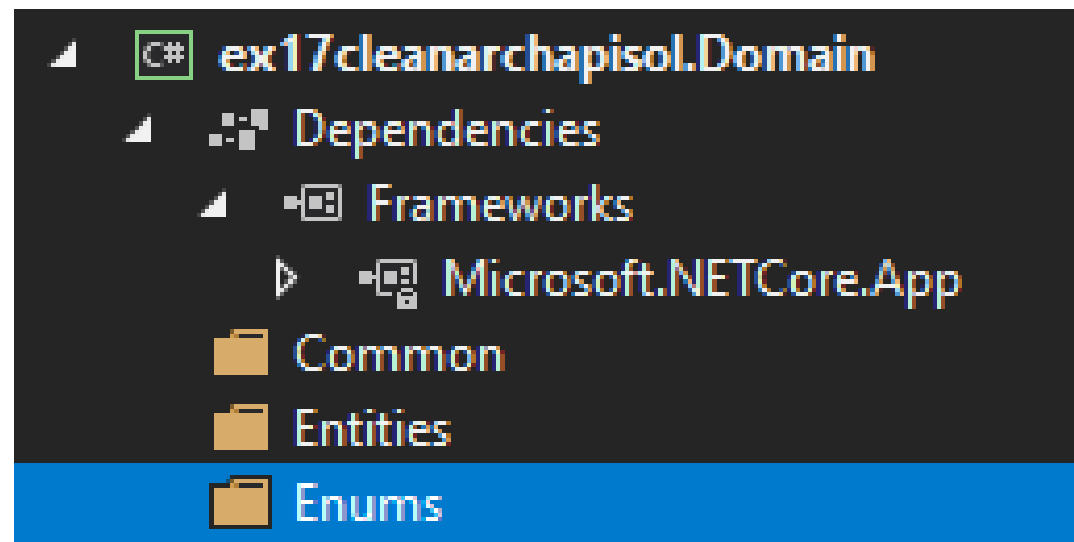
ex17cleanarchapisol.Infrastructure

ex17cleanarchapisol.WebApi



INSTALL PROJECT PACKAGE DEPENDENCIES

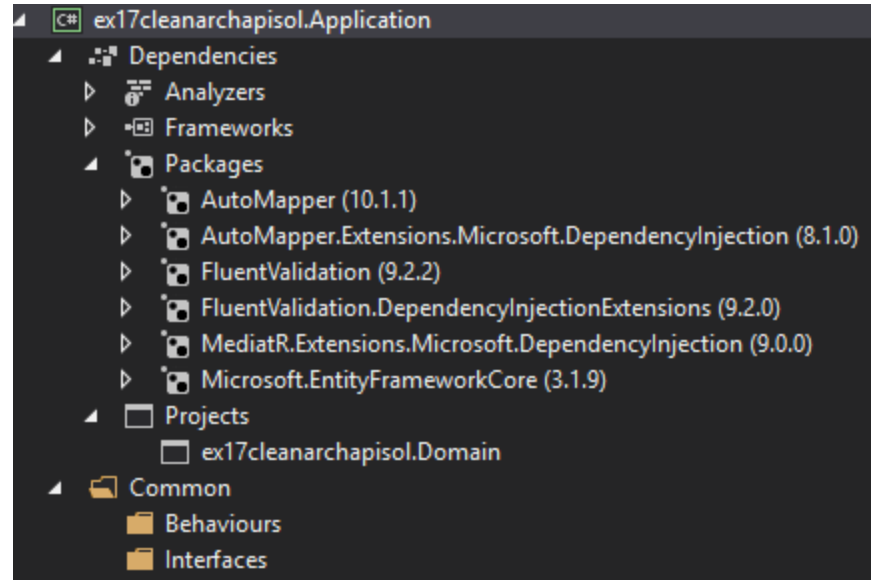
- Install the packages using Install-Package command at Package Manager Console.
- Add folders



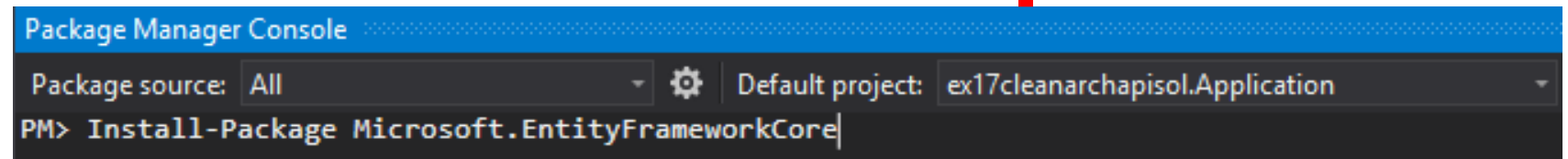
DOMAIN LAYER

ADD PROJECT REFERENCE

- Install the packages using Install-Package command at Package Manager Console.
- Add folders



INSTALL PROJECT PACKAGE DEPENDENCIES

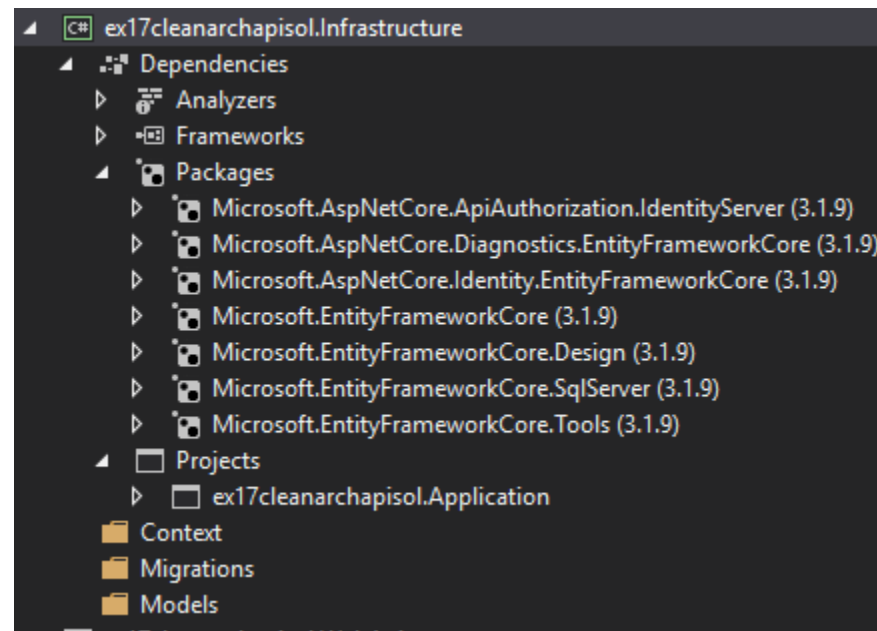


Application Layer

ADD PROJECT REFERENCE
DOMAIN LAYER

INSTALL PROJECT PACKAGE DEPENDENCIES

- Install the packages using Install-Package command at Package Manager Console.
- Add folders



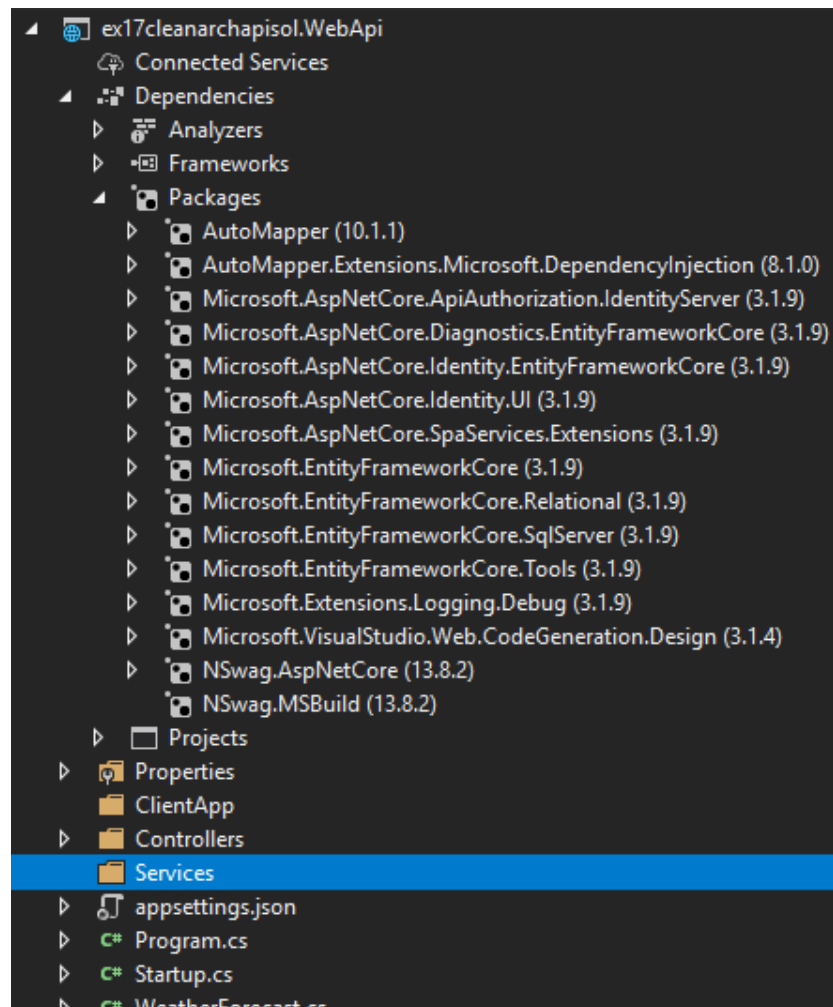
Install-Package

INFRASTRUCTURE LAYER

ADD PROJECT REFERENCE
APPLICATION LAYER

Example:

- Install the packages using Install-Package command at Package Manager Console.
- Add folders



WEBAPI LAYER

SET AS STARTUP PROJECT

INSTALL PROJECT PACKAGE DEPENDENCIES

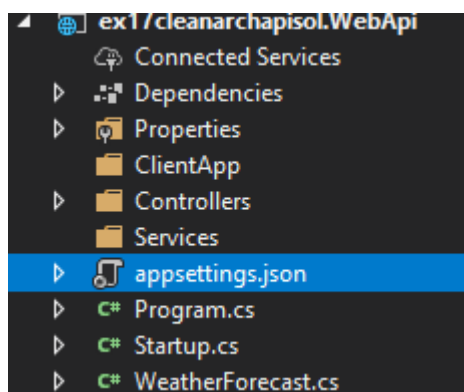
Install-Package

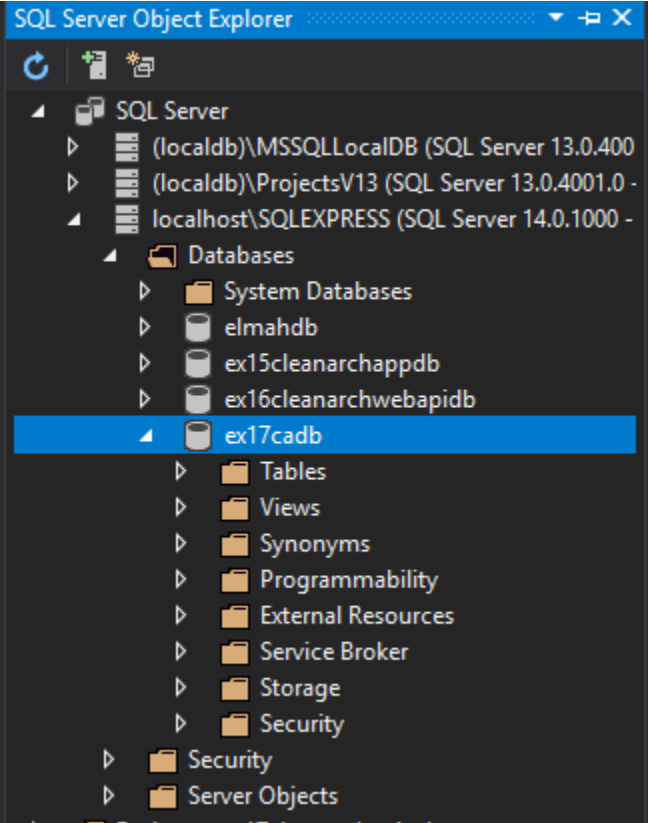
ADD PROJECT REFERENCE

INFRASTRUCTURE

▶ appsettings.json

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=localhost\\SQLEXPRESS; Database=ex17cadb; Trusted_Connection=True; MultipleActiveResultSets=true"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "IdentityServer": {
    "Key": {
      "Type": "Store",
      "StoreName": "My",
      "StoreLocation": "CurrentUser",
      "Name": "CN=**HistoryisABunchofLiesAgreedUponbypushingpeoplearound!**"
    }
  },
  "AllowedHosts": "*"
}
```



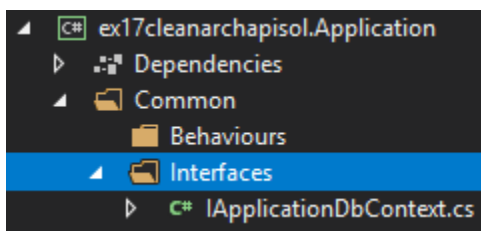


Application Layer

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading;
using System.Threading.Tasks;

namespace ex17cleanarchapisol.Application.Common.Interfaces
{
    0 references
    public interface IApplicationDbContext
    {
        //put the DbSets here

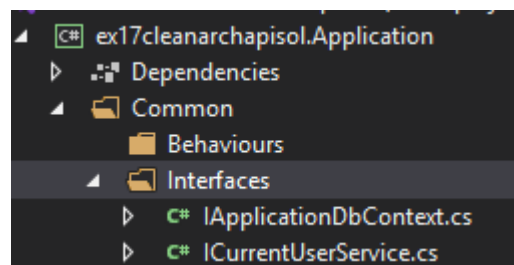
        0 references
        Task<int> SaveChangesAsync(CancellationToken cancellationToken);
    }
}
```

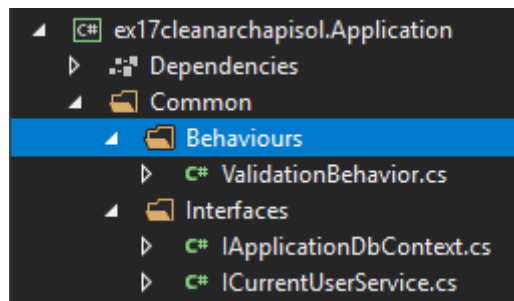


Application Layer

```
using System;
using System.Collections.Generic;
using System.Text;

namespace ex17cleanarchapisol.Application.Common.Interfaces
{
    0 references
    public interface ICurrentUserService
    {
        0 references
        string UserId { get; }
    }
}
```





```
using FluentValidation;
using MediatR;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;

namespace ex17cleanarchapisol.Application.Common.Behaviours
{
    1 reference
    public class ValidationBehavior<TRequest, TResponse> : IPipelineBehavior<TRequest, TResponse> where TRequest : IRequest<TResponse>
    {
        private readonly IEnumerable<IValidator<TRequest>> _validators;
        0 references
        public ValidationBehavior(IEnumerable<IValidator<TRequest>> validators)
        {
            _validators = validators;
        }
        0 references
        public Task<TResponse> Handle(TRequest request, CancellationToken cancellationToken, RequestHandlerDelegate<TResponse> next)
        {
            var context = new ValidationContext<TRequest>(request);

            //fluentapi version 8
            //var context = new ValidationContext(request);
            var failures = _validators.Select(v => v.Validate(context))
                                    .SelectMany(result => result.Errors)
                                    .Where(f => f != null)
                                    .ToList();

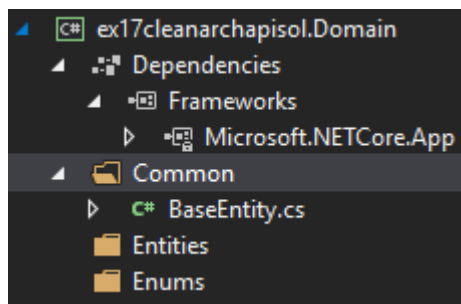
            if (failures.Count != 0)
            {
                throw new ValidationException(failures);
            }

            return next();
        }
    }
}
```

Domain Layer

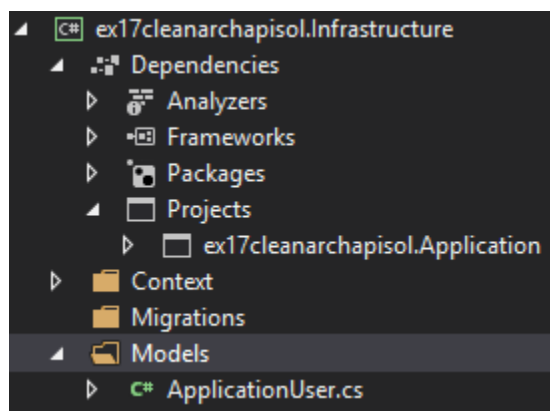
```
using System;
using System.Collections.Generic;
using System.Text;

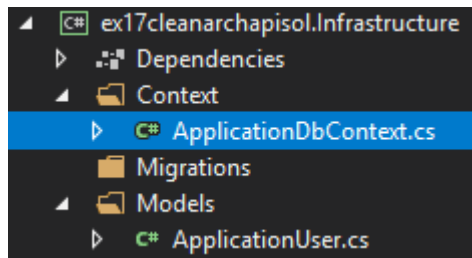
namespace ex17cleanarchapisol.Domain.Common
{
    0 references
    public class BaseEntity
    {
        0 references
        public string CreatedBy { get; set; }
        0 references
        public DateTime Created { get; set; }
        0 references
        public string LastModifiedBy { get; set; }
        0 references
        public DateTime? LastModified { get; set; }
    }
}
```



```
using Microsoft.AspNetCore.Identity;
using System;
using System.Collections.Generic;
using System.Text;

namespace ex17cleanarchapisol.Infrastructure.Models
{
    0 references
    public class ApplicationUser : IdentityUser
    {
        ---
    }
}
```





```
using ex17cleanarchapisol.Application.Common.Interfaces;
using ex17cleanarchapisol.Domain.Common;
using ex17cleanarchapisol.Infrastructure.Models;
using IdentityServer4.EntityFramework.Options;
using Microsoft.AspNetCore.ApiAuthorization.IdentityServer;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Options;
using System;
using System.Collections.Generic;
using System.Text;
using System.Threading;
using System.Threading.Tasks;

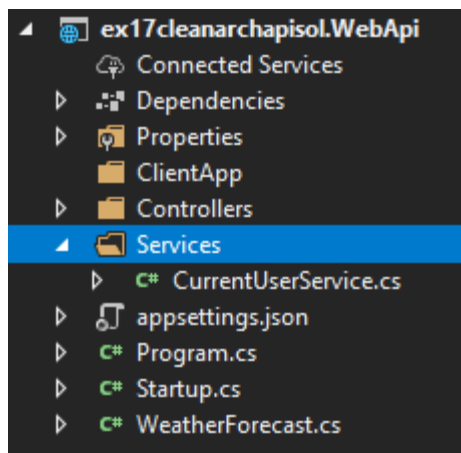
namespace ex17cleanarchapisol.Infrastructure.Context
{
    1 reference
    public class ApplicationDbContext : ApiAuthorizationDbContext<ApplicationUser>, IApplicationDbContext
    {
        private readonly ICurrentUserService _currentUserService;

        0 references
        public ApplicationDbContext(DbContextOptions options, IOptions<OperationalStoreOptions> operationalStoreOptions, ICurrentUserService currentUserService) : base(options, operationalStoreOptions)
        {
            _currentUserService = currentUserService;

            //insert the DbSet here
            //public DbSet<EntityName> PluralEntity{get;set;}

            1 reference
            public override Task<int> SaveChangesAsync(CancellationToken cancellationToken)
            {
                foreach (var entry in ChangeTracker.Entries<BaseEntity>())
                {
                    switch (entry.State)
                    {
                        case EntityState.Added:
                            entry.Entity.CreatedBy = _currentUserService.UserId;
                            entry.Entity.Created = DateTime.UtcNow;
                            break;
                        case EntityState.Modified:
                            entry.Entity.LastModifiedBy = _currentUserService.UserId;
                            entry.Entity.LastModified = DateTime.UtcNow;
                            break;
                    }
                }

                return base.SaveChangesAsync(cancellationToken);
            }
        }
    }
}
```

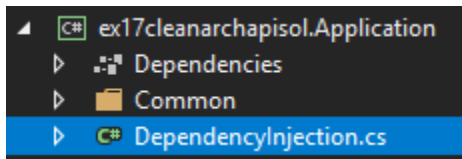



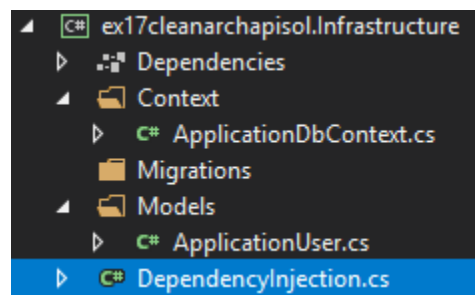
```
using ex17cleanarchapisol.Application.Common.Interfaces;
using Microsoft.AspNetCore.Http;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Claims;
using System.Threading.Tasks;

namespace ex17cleanarchapisol.WebApi.Services
{
    1 reference
    public class CurrentUserService : ICurrentUserService
    {
        0 references
        public CurrentUserService(IHttpContextAccessor httpContextAccessor)
        {
            this.UserId = httpContextAccessor.HttpContext?.User?.FindFirstValue
                (ClaimTypes.NameIdentifier);
        }
        4 references
        public string UserId { get; }
    }
}
```

```
using AutoMapper;
using ex17cleanarchapisol.Application.Common.Behaviours;
using FluentValidation;
using MediatR;
using Microsoft.Extensions.DependencyInjection;
using System;
using System.Collections.Generic;
using System.Reflection;
using System.Text;

namespace ex17cleanarchapisol.Application
{
    0 references
    public static class DependencyInjection
    {
        1 reference
        public static IServiceCollection AddApplication(this IServiceCollection services)
        {
            services.AddMediatR(Assembly.GetExecutingAssembly());
            services.AddTransient(typeof(IPipelineBehavior<, >), typeof(ValidationBehavior<, >));
            services.AddValidatorsFromAssembly(Assembly.GetExecutingAssembly());
            services.AddAutoMapper(Assembly.GetExecutingAssembly());
            return services;
        }
    }
}
```





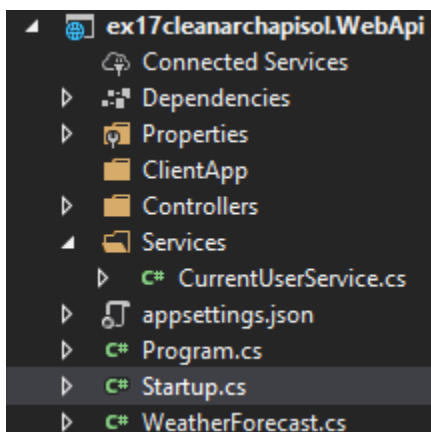
```
using ex17cleanarchapisol.Application.Common.Interfaces;
using ex17cleanarchapisol.Infrastructure.Context;
using ex17cleanarchapisol.Infrastructure.Models;
using Microsoft.AspNetCore.Authentication;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using System;
using System.Collections.Generic;
using System.Text;

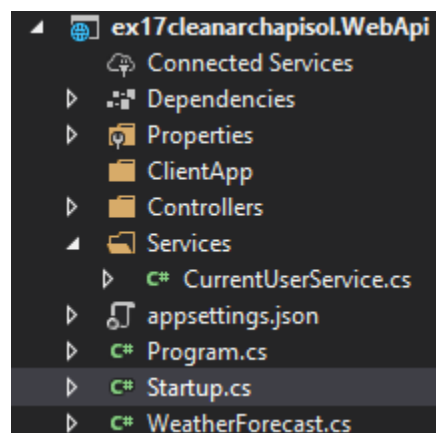
namespace ex17cleanarchapisol.Infrastructure
{
    0 references
    public static class DependencyInjection
    {
        1 reference
        public static IServiceCollection AddInfrastructure(this IServiceCollection
            services, Microsoft.Extensions.Configuration.IConfiguration configuration)
        {
            services.AddDbContext<ApplicationDbContext>(options =>
                options.UseSqlServer(
                    configuration.GetConnectionString("DefaultConnection"),
                    b => b.MigrationsAssembly(typeof(
                        ApplicationDbContext).Assembly.FullName));
            services.AddScoped<IApplicationDbContext>(provider =>
                provider.GetService<ApplicationDbContext>());
            services.AddDefaultIdentity<ApplicationUser>(options =>
                options.SignIn.RequireConfirmedAccount = true)
                .AddEntityFrameworkStores<ApplicationDbContext>();
            services.AddIdentityServer()
                .AddApiAuthorization<ApplicationUser, ApplicationDbContext>();
            services.AddAuthentication()
                .AddIdentityServerJwt();
            return services;
        }
    }
}
```

Startup.cs

```
0 references
public void ConfigureServices(IServiceCollection services)
{
    //application layer
    services.AddApplication();
    //infrastructure layer
    services.AddInfrastructure(Configuration);
    //register the services
    services.AddScoped<ICurrentUserService, CurrentUserService>();
    //controllers
    services.AddControllers();
    // In production, the React files will be served from this directory
    services.AddSpaStaticFiles(configuration =>
    {
        configuration.RootPath = "ClientApp/build";
    });
    //open api configuration
    services.AddOpenApiDocument(configure =>
    {
        configure.Title = "CleanArchitecture API Template";
        configure.AddSecurity("JWT", Enumerable.Empty<string>(), new
            OpenApiSecurityScheme
            {
                Type = OpenApiSecuritySchemeType.ApiKey,
                Name = "Authorization",
                In = OpenApiSecurityApiKeyLocation.Header,
                Description = "Type into the textbox: Bearer {your JWT token}."
            });

        configure.OperationProcessors.Add(new
            AspNetCoreOperationSecurityScopeProcessor("JWT"));
    });
    // Register the Swagger services
    services.AddSwaggerDocument();
}
```





```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
        app.UseDatabaseErrorPage();
        app.UseOpenApi();
        app.UseSwaggerUi3();
    }

    app.UseHttpsRedirection();
    app.UseStaticFiles();
    app.UseSpaStaticFiles();

    app.UseRouting();

    app.UseAuthorization();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllers();
    });
}
```

Package Manager Console

Package source: All Default project: ex17cleanarchapisol.Infrastructure

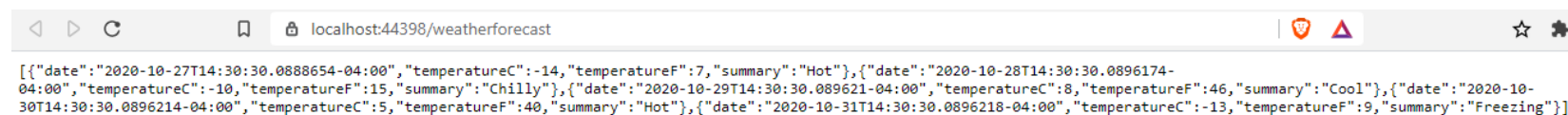
```
PM> Add-Migration InitialCreate
Build started...
Build succeeded.
To undo this action, use Remove-Migration.
PM> update-database
Build started...
Build succeeded.
Done.
```

SQL Server Object Explorer

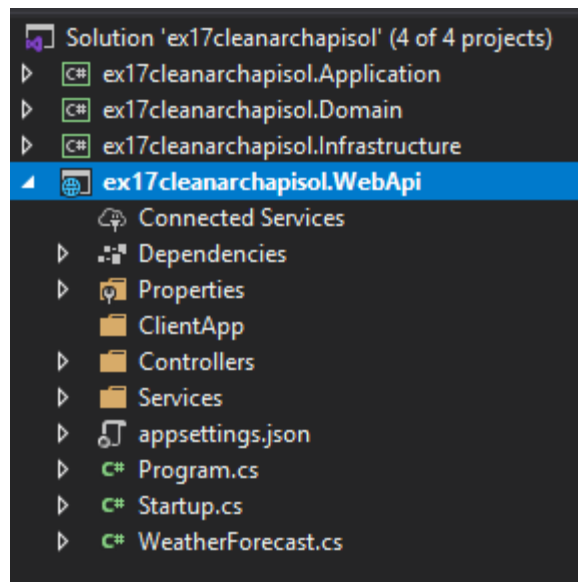
- SQL Server
 - (localdb)\MSSQLLocalDB (SQL Server 13.0.4001)
 - (localdb)\ProjectsV13 (SQL Server 13.0.4001)
 - localhost\SQLEXPRESS (SQL Server 14.0.10000)
 - Databases
 - System Databases
 - elmahdb
 - ex15cleanarchappdb
 - ex16cleanarchwebapidb
 - ex17cadb
 - Tables
 - System Tables
 - FileTables
 - External Tables
 - dbo.__EFMigrationsHistory
 - dbo.AspNetRoleClaims
 - dbo.AspNetRoles
 - dbo.AspNetUserClaims
 - dbo.AspNetUserLogins
 - dbo.AspNetUserRoles
 - dbo.AspNetUsers
 - dbo.AspNetUserTokens
 - dbo.DeviceCodes
 - dbo.PersistedGrants
 - Views
 - Synonyms
 - Programmability

- ex17cleanarchapisol.Infrastructure
 - Dependencies
 - Context
 - ApplicationDbContext.cs
 - Migrations
 - 20201026181315_InitialCreate.cs
 - 20201026181315_InitialCreate.Designer.cs
 - ApplicationDbContextModelSnapshot.cs
 - Models
 - ApplicationUser.cs
 - DependencyInjection.cs

<https://localhost:44398/weatherforecast>



```
[{"date": "2020-10-27T14:30:30.0888654-04:00", "temperatureC": -14, "temperatureF": 7, "summary": "Hot"}, {"date": "2020-10-28T14:30:30.0896174-04:00", "temperatureC": -10, "temperatureF": 15, "summary": "Chilly"}, {"date": "2020-10-29T14:30:30.089621-04:00", "temperatureC": 8, "temperatureF": 46, "summary": "Cool"}, {"date": "2020-10-30T14:30:30.0896214-04:00", "temperatureC": 5, "temperatureF": 40, "summary": "Hot"}, {"date": "2020-10-31T14:30:30.0896218-04:00", "temperatureC": -13, "temperatureF": 9, "summary": "Freezing"}]
```



- We have successfully created an CleanArchitecture –WebAPI Template using CQRS/MediatR + AutoMapper
- We can use this template, for our future project demos!