Add migration with different assembly

Asked 3 years, 8 months ago Active 1 month ago Viewed 34k times

40

12

I am doing a project in ASP.NET CORE 1.0.0 and I am using EntityFrameworkCore. I have separate assemblies and my project structure looks like this:

```
ProjectSolution
   -src
      -1 Domain
         -Project.Data
      -2 Api
         -Project.Api
```

In my `Project.Api` is the `Startup` class

```
public void ConfigureServices(IServiceCollection services)
    {
        services.AddDbContext<ProjectDbContext>();

        services.AddIdentity<IdentityUser, IdentityRole>()
                .AddEntityFrameworkStores<ProjectDbContext>()
                .AddDefaultTokenProviders();
    }
```

The `DbContext` is in my `Project.Data` project

```
public class ProjectDbContext : IdentityDbContext<IdentityUser>
{
    public ProjectDbContext(DbContextOptions<ProjectDbContext> options) : base(options)
    {

    }

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {

        var builder = new ConfigurationBuilder();
        builder.SetBasePath(Directory.GetCurrentDirectory());
        builder.AddJsonFile("appsettings.json");
        IConfiguration Configuration = builder.Build();

        optionsBuilder.UseSqlServer(
            Configuration.GetConnectionString("DefaultConnection"));
```

When I try to make the initial migration, I get this error:

> "Your target project 'Project.Api' doesn't match your migrations assembly 'Project.Data'. Either change your target project or change your migrations assembly. Change your migrations assembly by using DbContextOptionsBuilder. E.g. options.UseSqlServer(connection, b => b.MigrationsAssembly("Project.Api")). By default, the migrations assembly is the assembly containing the DbContext. Change your target project to the migrations project by using the Package Manager Console's Default project drop-down list, or by executing "dotnet ef" from the directory containing the migrations project."

After I seeing this error, I tried to execute this command located in `Project.Api` :

> dotnet ef --startup-project ../Project.Api --assembly "../../1 Data/Project.Data" migrations add Initial

and I got this error:

> "Unexpected value '../../1 Domain/Project.Data' for option 'assembly'"

I don't know why I get this error, when I try to execute the command with the '-assembly' parameter.

I can't create a Initial Migration from other assembly and I've searched for information about it but didn't got any results.

Has someone had similar issues?

c# asp.net-core entity-framework-core ef-migrations

edited Oct 12 '19 at 9:26
mark_h
3,504 3 26 36

asked Aug 1 '16 at 18:29
kdar
431 1 5 8

Did you read the docs? docs.efproject.net/en/latest/miscellaneous/cli/… – Tseng Aug 1 '16 at 18:36

Yes, but i didn t `get fix the error, i tryed diferent options of commands with --startup-project and --assebly but i didn` t get nothing – kdar Aug 1 '16 at 20:01

Don't use --assembly. That's an internal implementation detail that should have been hidden from the help message, but shows up anyways because of github.com/aspnet/EntityFramework/issues/5188 – natemcmaster Aug 2 '16 at 16:10

11 Answers

Active Oldest Votes

Não encontrou uma resposta? Pergunte em Stack Overflow em Português. ×

63

```
if (targetAssembly != migrationsAssembly)
       throw MigrationsAssemblyMismatchError;
```

targetAssembly = the target project you are operating on. On the command line, it is the project in the current working directory. In Package Manager Console, it is whatever project is selected in the drop down box on the top right of that window pane.

migrationsAssembly = assembly containing code for migrations. This is configurable. By default, this will be the assembly containing the DbContext, in your case, Project.Data.dll. As the error message suggests, you have have a two options to resolve this

1 - Change target assembly.

```
cd Project.Data/
dotnet ef --startup-project ../Project.Api/ migrations add Initial

// code doesn't use .MigrationsAssembly...just rely on the default
options.UseSqlServer(connection)
```

2 - Change the migrations assembly.

```
cd Project.Api/
dotnet ef migrations add Initial

// change the default migrations assembly
options.UseSqlServer(connection, b => b.MigrationsAssembly("Project.Api"))
```

edited Aug 2 '16 at 17:19

answered Aug 2 '16 at 16:09
natemcmaster
18.9k 2 61 93

1 I can `t execute a command from Project.Data because is a classlibrary and it doesn` t allow execute command, i only can execute command in Project.Api – kdar Aug 3 '16 at 5:43

Thanks , it was very useful ;) – Arash Nov 20 '17 at 12:18

1 Thanks, this helped after spending a lot of time on other solutions. – Pawan Pillai Mar 13 '18 at 16:20

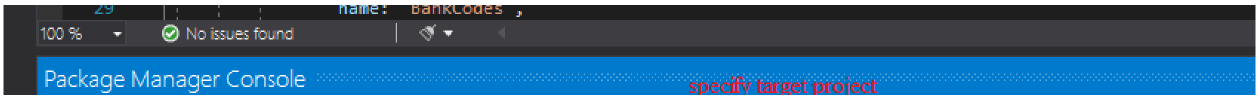
Excellent explanation and examples, solved my problem immediately after wasting hours on other "solutions". Thank you!!! – Cydaps Oct 31 '18 at 16:43

1 Do you have any explanation or link to one as to why this check was implemented? What was the reason behind forbidding migrations when the context is from an other assembly? – Mathieu VIALES Feb 2 '19 at 14:01

30

I had the same problem until I noticed that on the package manager console top bar => "Default Projects" was supposed to be "Project.Data" and not "Project.API".

Once you target the "Project.Data" from the dropdown list and run the migration you should be fine.



edited Oct 12 '19 at 18:24
Oge Nwike
191 2 8

answered Dec 8 '17 at 0:42
Dre Ross
407 4 6

1 You saved my day!:) – StepUp Dec 28 '19 at 13:26

12

Using EF Core 2, you can easily separate your *Web* project from your *Data* (DbContext) project. In fact, you just need to implement the *IDesignTimeDbContextFactory* interface. According to [Microsoft docs](), *IDesignTimeDbContextFactory* is:

> A factory for creating derived DbContext instances. Implement this interface to enable design-time services for context types that do not have a public default constructor. At design-time, derived DbContext instances can be created in order to enable specific design-time experiences such as Migrations. Design-time services will automatically discover implementations of this interface that are in the startup assembly or the same assembly as the derived context.

In the bottom code snippet you can see my implementation of DbContextFactory which is defined inside my *Data* project:

```
public class DbContextFactory : IDesignTimeDbContextFactory<KuchidDbContext>
{
    public KuchidDbContext CreateDbContext(string[] args)
    {
        var configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("appsettings.json")
            .Build();

        var dbContextBuilder = new DbContextOptionsBuilder<KuchidDbContext>();

        var connectionString = configuration.GetConnectionString("Kuchid");

        dbContextBuilder.UseSqlServer(connectionString);

        return new KuchidDbContext(dbContextBuilder.Options);
    }
}
```

Now, I can initialize EF migration by setting my *Web* project as the StartUp project and selecting my *Data* project inside the Package Manager Console.

```
Add-Migration initial
```

You can find more details [here](). However, this blog post uses an obsoleted class instead of *IDesignTimeDbContextFactory*.

answered May 7 '18 at 4:59
Ehsan Mirsaeedi
3,322 24 29

you cannot run the "migrations add" command because it would attempt to add the migrations to where the db context is. You can only run "ef database update" and "ef database drop" in this fashion. – Ales Potocnik Hahonina Feb 18 '19 at 16:57

1 You have used `.AddJsonFile("appsettings.json")` . What if appsettings.json file do not exist in your .Data Project? – Himalaya Garg Jun 2 '19 at 7:22

7

(ASP.NET Core 2+)

Had the same issue. Here is what I did:

1. Reference the project that contains the DbContext (Project.A) from the project that will contain the migrations (Project.B).
2. Move the existing migrations from Project.A to Project.B (If you don't have migrations - create them first)
3. Configure the migrations assembly inside Project.A

options.UseSqlServer(connectionString, x => x.MigrationsAssembly("Project.B"));

Assuming your projects reside in the same parent folder:

4. `dotnet ef migrations add Init --p Project.B -c DbContext`

The migrations now go to Project.B

Source: Microsoft

edited Dec 22 '19 at 10:24

answered Mar 25 '19 at 21:09
Kaloyan Drenski
612 1 9 15

I was missing the -c DbContext. Thanks! – Thomas Vincent Blomberg Oct 13 '19 at 19:13

5

I ran on the same problem and found this

We're you trying to run your migrations on a class library? So was I. Turns out this isn't supported yet, so we'll need to work around it.

EDIT: I found solution on this git repo

edited Mar 12 '17 at 19:07

answered Aug 5 '16 at 18:54
Alan Jagar
171 12

Thank you. I saw this solution but it seembed a little forced but so far it is the only solution I've seen it work. i will follow the link to get the news – kdar Aug 6 '16 at 12:26

Thanks a lot!! services.AddDbContext<EventbookingContext>(options => options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection"), x => x.MigrationsAssembly("ClasslibraryGoesHere"))); – NetProvoke May 31 '17 at 19:59

4

And just **side note** for anybody else who wants to add migrations to specific folder inside your project:

EF CLI not support this yet. I tried `--data-dir` but it didn't work.

The only thing works is to use Package Manager Console:

1. Pick your default project
2. use `-OutputDir` command parameter, .e.g., `Add-Migration InitConfigurationStore -OutputDir PersistedStores/ConfigurationStore` command will output the mgiration to the folder 'PersistedStores/ConfigurationStore' in my project.

Updates as of 10/12/2017

```
public void ConfigureServices(IServiceCollection services)
{
    ...

    string dbConnectionString = services.GetConnectionString("YOUR_PROJECT_CONNECTION");
    string assemblyName = typeof(ProjectDbContext).Namespace;

    services.AddDbContext<ProjectDbContext>(options =>
        options.UseSqlServer(dbConnectionString,
            optionsBuilder =>
                optionsBuilder.MigrationsAssembly(assemblyName)
        )
   );

   ...
}
```

edited Oct 12 '17 at 21:13

answered Sep 21 '16 at 16:03
David Liang
10.3k 2 27 46

2

There are multiple projects included in the Solution.

Solution
|- MyApp (Startup Proj)
|- MyApp.Migrations (ClassLibrary)

```
Add-Migration NewMigration -Project MyApp.Migrations
```

Note: MyApp.Migrations also includes the DbContext.

edited Dec 15 '19 at 8:16
Ariel
1,273 1 15 25

answered Mar 28 '19 at 21:04
Majid joghataey
661 8 21

2 Can you explain the context of this more to make it a better answer. – danblack Mar 28 '19 at 22:07

This was helpful, thank you. I placed my Migrations folder and DbContext in a class library and then reference the class library in my asp.net web app project. The reason is that I am also using the class

1

Add Migration With CLI Command:

```
dotnet ef migrations add NewMigration --project YourAssemplyName
```

Add Migration With PMC Command:

```
Add-Migration NewMigration -Project YourAssemplyName
```

answered Dec 22 '19 at 10:34
AminGolmahalle
1,095 10 16

0

I was facing similar issue, though answers seems straight forward somehow they didn't work. My Answer is similar to @Ehsan Mirsaeedi, with small change in DbContextFactory class. Instead of Adding migration assembly name in Startup class of API, I have mentioned in DbContextFactory class which is part of Data project(class library).

```
public class DbContextFactory : IDesignTimeDbContextFactory<KuchidDbContext>
{
   public KuchidDbContext CreateDbContext(string[] args)
   {
       var configuration = new ConfigurationBuilder()
          .SetBasePath(Directory.GetCurrentDirectory())
          .AddJsonFile("appsettings.json")
          .Build();

       var dbContextBuilder = new DbContextOptionsBuilder<KuchidDbContext>();

       var connectionString = configuration.GetConnectionString("connectionString");

       var migrationAssemblyName=
configuration.GetConnectionString("migrationAssemblyName");

       dbContextBuilder.UseSqlServer(connectionString, o =>
o.MigrationAssembly(migrationAssemblyName));

       return new KuchidDbContext(dbContextBuilder.Options);
   }
}
```

You would need 'Microsoft.Extensions.Configuration' and 'Microsoft.Extensions.Configuration.Json' for SetBasePath & AddJsonFile extensions to work.

Note: I feel this is just a work around. It should pickup the DbContextOptions from the startup class somehow it is not. I guess there is definitely some wiring issue.

answered Feb 26 '19 at 1:39
Mady
323 4 17

I have resolved it by adding below line in Startup.cs. Hope it will help you also. I have used

```
     var migrationsAssembly = typeof(Startup).GetTypeInfo().Assembly.GetName().Name;
services.AddIdentityServer(options =>
            {
                options.Events.RaiseErrorEvents = true;
                options.Events.RaiseInformationEvents = true;
                options.Events.RaiseFailureEvents = true;
                options.Events.RaiseSuccessEvents = true;

            })
                .AddSigningCredential(cert)
                 .AddCustomUserStore<IdentityServerConfigurationDbContext>()
                // this adds the config data from DB (clients, resources)
                .AddConfigurationStore(options =>
                {
                    options.ConfigureDbContext = builder =>
                        builder.UseNpgsql(connectionString,
                            sql => sql.MigrationsAssembly(migrationsAssembly));
                })
                // this adds the operational data from DB (codes, tokens, consents)
                .AddOperationalStore(options =>
                {
                    options.ConfigureDbContext = builder =>
                        builder.UseNpgsql(connectionString,
                            sql => sql.MigrationsAssembly(migrationsAssembly));

                    // this enables automatic token cleanup. this is optional.
                    options.EnableTokenCleanup = true;
                    options.TokenCleanupInterval = 30;
                });
```

answered Dec 24 '19 at 5:31
MayankGaur
554 3 19

-1

For all of you who have ***multiple startup projects***.

Notice that you need to set your target project as startup project - `Project.Api` (form the question example) should be the startup project.

Hope that will help someone :)

answered Jul 12 '18 at 11:35
Atanas Sarafov
9 4

Dependency Injection requires a single startup project. – RandyDaddis Apr 16 '19 at 18:09