` 1

Scaffolding DbContext and Models with EntityFramework Core 2.0 and the CLI

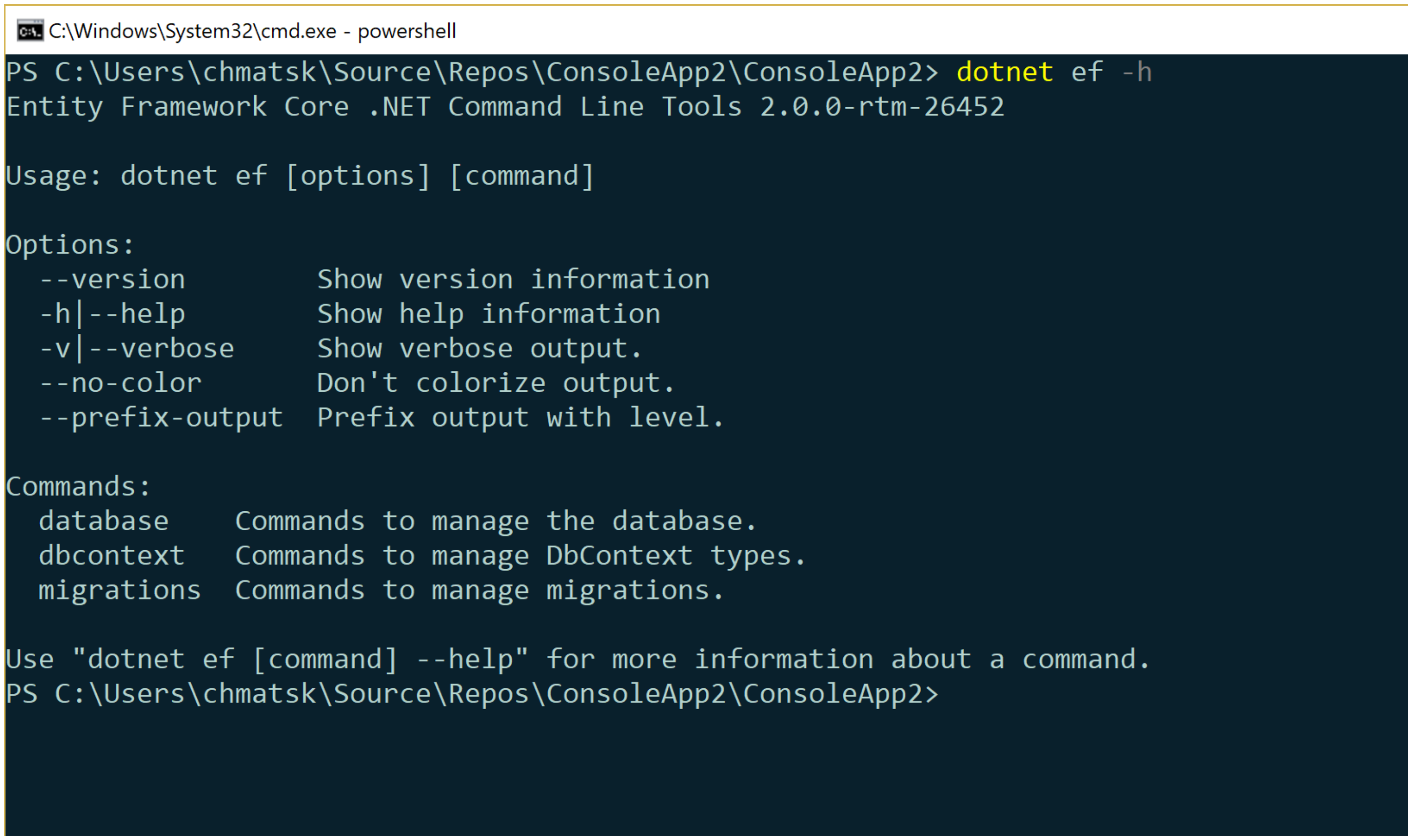
07 September 2017 **Comments (/scaffolding-dbcontext-and-models-with-entityframework-core-2-0-and-the-cli/#disqus_thread)** Posted in .NET Core (/tag/net-core/), ef core (/tag/ef-core/), command line (/tag/command-line/), CLI (/tag/cli/)

EF Core 2.0 has been out for a few weeks now. If you're looking at an ORM for your .NET application then EF Core should be at the top of the list of possible options. I say at the top and not the only one because depending on the project requirements, some features may be missing. For example, EF Core cut ties with `.edmx` so if you want to stick with the designer feature you will need to use EF6. There are other limitations so make sure you have a look at this post (https://docs.microsoft.com/en-us/ef/core/miscellaneous/1x-2x-upgrade) for API changes in 2.0 and also at the list of providers that may or may not be supported yet.

If EF Core fits your needs, then you're up for a treat, considering that it comes with significant performance improvements, smaller footprint and a lot of cool features to help you get your Data Access Layer (DAL) set up quickly. In most cases, you will use the Code-First approach, especially if it's a new project with no back-end. There are though those cases that you may already have a database in place. You don't want to hand-roll your models, definitely not! So how would you do it?

USING THE EFCORE TOOLS AND CLI TO SCAFFOLD YOUR CODE

EF Core comes with an awesome CLI that can help you manage many aspects of EF Core in your project. You can have a look at all the available CLI options in the official docs - here (https://docs.microsoft.com/en-us/ef/core/miscellaneous/cli/dotnet).


C:\Windows\System32\cmd.exe - powershell

```
PS C:\Users\chmatsk\Source\Repos\ConsoleApp2\ConsoleApp2> dotnet ef -h
Entity Framework Core .NET Command Line Tools 2.0.0-rtm-26452

Usage: dotnet ef [options] [command]

Options:
  --version        Show version information
  -h|--help        Show help information
  -v|--verbose     Show verbose output.
  --no-color       Don't colorize output.
  --prefix-output  Prefix output with level.

Commands:
  database    Commands to manage the database.
  dbcontext   Commands to manage DbContext types.
  migrations  Commands to manage migrations.

Use "dotnet ef [command] --help" for more information about a command.
PS C:\Users\chmatsk\Source\Repos\ConsoleApp2\ConsoleApp2>
```

The EF Core tools run in the context of the `dotnet` CLI.

> Note: At the time of writing this post there's a known with support for .NETStandard 2.0. To work around it, you'll need to multitarget against .NET Core 2.0 or the full .NET Framework to allow the CLI commands to execute on your project correctly.

To scaffold an existing database you need to ensure that you've installed the appropriate NuGet packages. I'm using a Console App for this example so I don't have to worry about the issue I described above with NETStandard 2.0. On you project, you can either use the NuGet package manager or fire up the command line, navigate to your project and type the following commands:

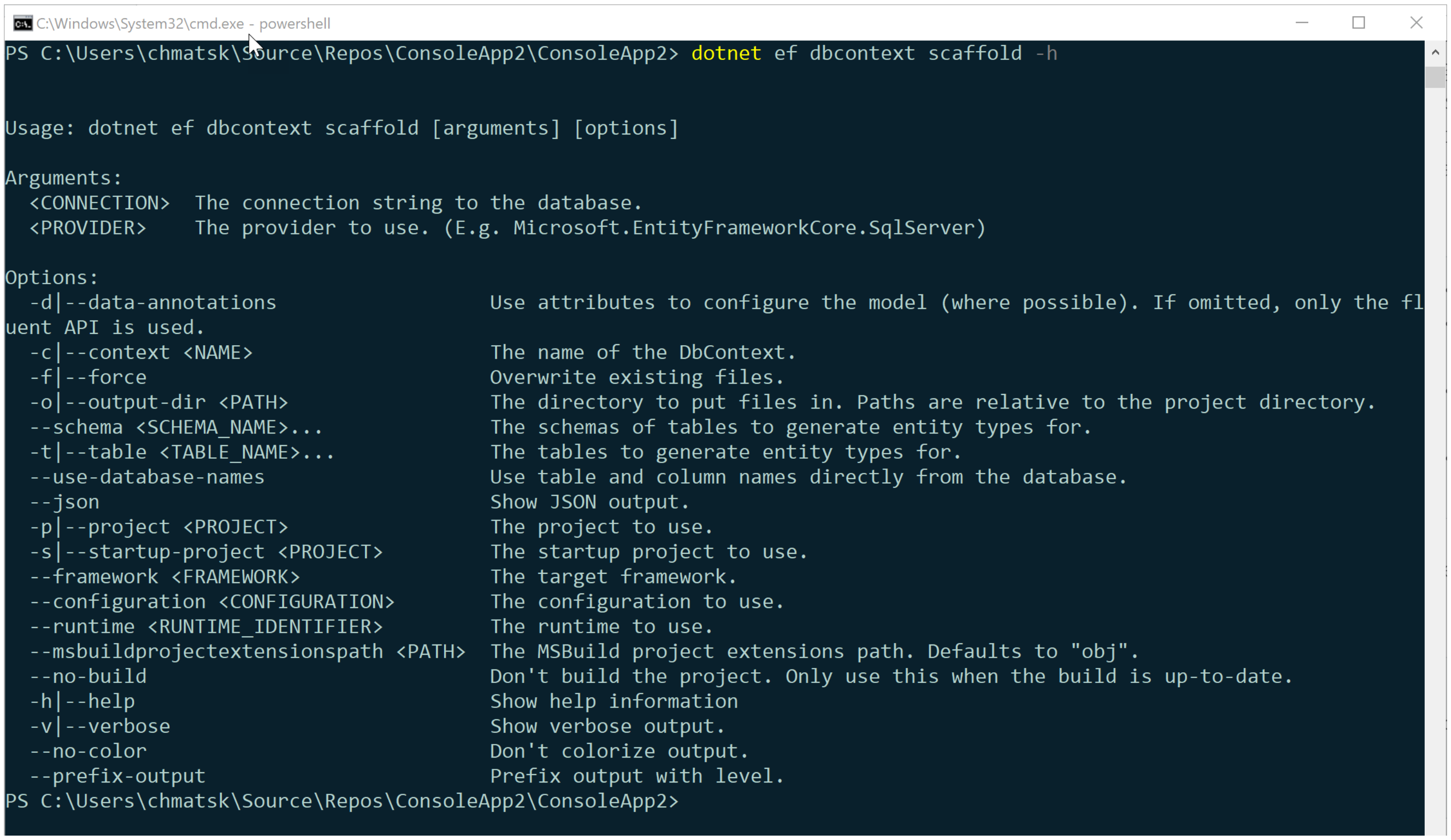
```
dotnet add package Microsoft.EntityFrameworkCore -v 2.0.0
dotnet add package Microsoft.EntityFrameworkCore.Design-v 2.0.0
dotnet add package Microsoft.EntityFrameworkCore.SqlServer-v 2.0.0
dotnet add package Microsoft.EntityFrameworkCore.Tools.DotNet-v 2.0.0
```

We also need to edit the *.csproj file to enable the CLI tool integration. Open the `.csproj` file and add the following:

```
<ItemGroup>
  <DotNetCliToolReference Include="Microsoft.EntityFrameworkCore.Tools.DotNet" Version="2.0.0" />
</ItemGroup>
```

Now we're ready to scaffold our code. In the command line, you can see the full list of options available when scaffolding from an existing database using the following command:

`dotnet ef dbcontext scaffold -h`



```
C:\Windows\System32\cmd.exe - powershell
PS C:\Users\chmatsk\Source\Repos\ConsoleApp2\ConsoleApp2> dotnet ef dbcontext scaffold -h


Usage: dotnet ef dbcontext scaffold [arguments] [options]

Arguments:
  <CONNECTION>  The connection string to the database.
  <PROVIDER>    The provider to use. (E.g. Microsoft.EntityFrameworkCore.SqlServer)

Options:
  -d|--data-annotations                  Use attributes to configure the model (where possible). If omitted, only the fl
uent API is used.
  -c|--context <NAME>                    The name of the DbContext.
  -f|--force                             Overwrite existing files.
  -o|--output-dir <PATH>                 The directory to put files in. Paths are relative to the project directory.
  --schema <SCHEMA_NAME>...              The schemas of tables to generate entity types for.
  -t|--table <TABLE_NAME>...             The tables to generate entity types for.
  --use-database-names                   Use table and column names directly from the database.
  --json                                 Show JSON output.
  -p|--project <PROJECT>                 The project to use.
  -s|--startup-project <PROJECT>         The startup project to use.
  --framework <FRAMEWORK>                The target framework.
  --configuration <CONFIGURATION>        The configuration to use.
  --runtime <RUNTIME_IDENTIFIER>         The runtime to use.
  --msbuildprojectextensionspath <PATH>  The MSBuild project extensions path. Defaults to "obj".
  --no-build                             Don't build the project. Only use this when the build is up-to-date.
  -h|--help                              Show help information
  -v|--verbose                           Show verbose output.
  --no-color                             Don't colorize output.
  --prefix-output                        Prefix output with level.
PS C:\Users\chmatsk\Source\Repos\ConsoleApp2\ConsoleApp2>
```

An example I used to scaffold an AdventureWorks database deployed on Azure SQL is provided below:

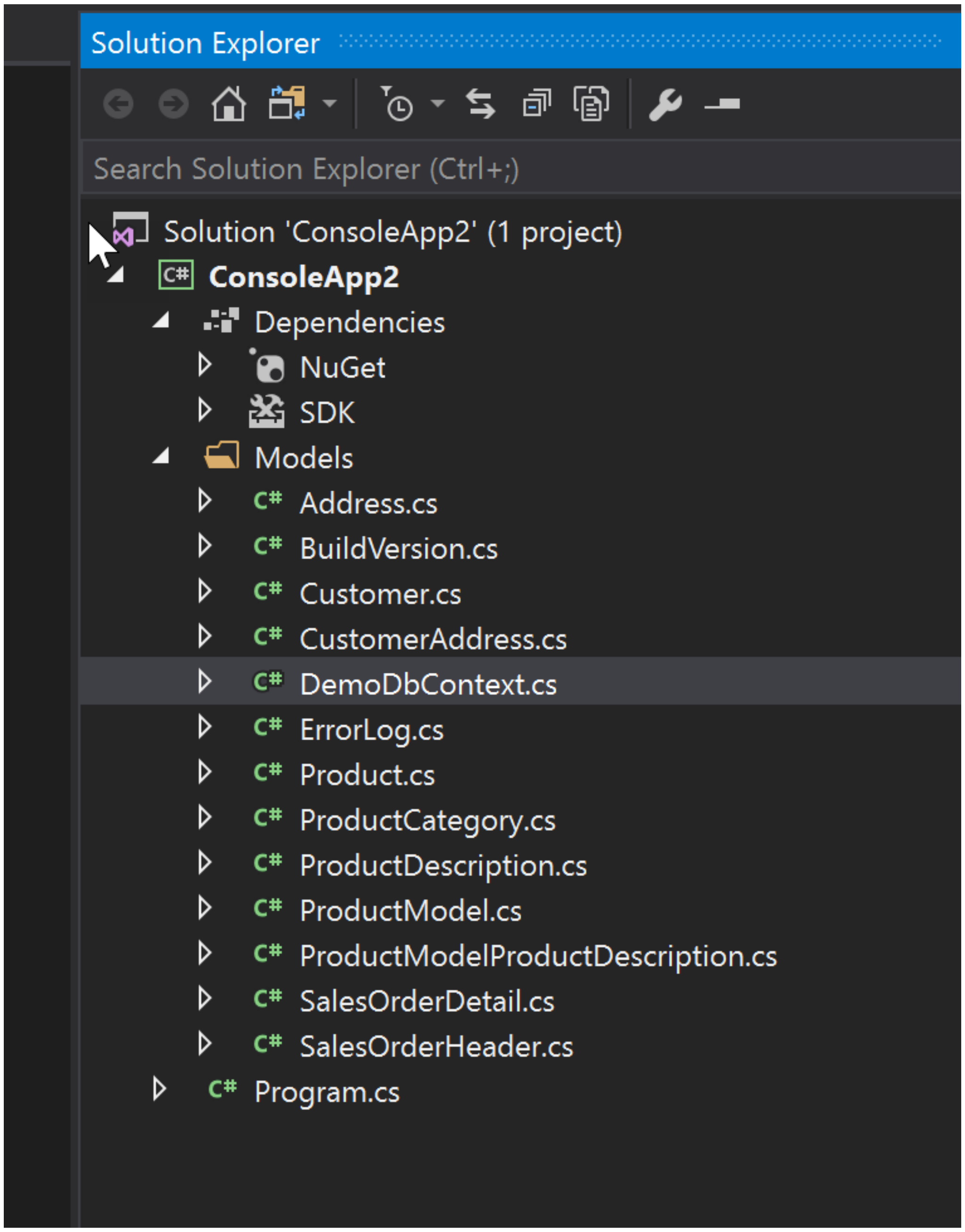
```
dotnet ef dbcontext scaffold "Server=tcp:<myAzureSQLServer>.databa
se.windows.net,1433;Initial Catalog=<MyDB>;Persist Security Info=False;User ID=<myDBAdmin>;Password=<myDB
Password>;Multipl
eActiveResultSets=False;Encrypt=True;TrustServerCertificate=False;Connection Timeout=30;" Microsoft.Entit
yFrameworkCore.
SqlServer -o Models -f -c DemoDbContext
```

Dissecting the command:

- ConnectionString (the Azure SQL connection string)
- Provider (sql server in this instance)

- `-o` for the output directory (Models)
- `-f` to override previous auto-generated code
- `-c` the name of the DbContext to use in the application

This resulted in the following code inside my project:



```
Solution 'ConsoleApp2' (1 project)
  ConsoleApp2
    Dependencies
      NuGet
      SDK
    Models
      Address.cs
      BuildVersion.cs
      Customer.cs
      CustomerAddress.cs
      DemoDbContext.cs
      ErrorLog.cs
      Product.cs
      ProductCategory.cs
      ProductDescription.cs
      ProductModel.cs
      ProductModelProductDescription.cs
      SalesOrderDetail.cs
      SalesOrderHeader.cs
    Program.cs
```

At this point I can instantiate the `DemoDbContext` in the application and start interacting with the database. As you can see, it took a couple of commands to install the NuGet packages, expose the EF CLI tools to the `dotnet` CLI context and scaffold the code from the existing codebase.

Pretty awesome if you ask me :)

P.S. Make sure you follow me on Twitter @christosmatskas (https://twitter.com/christosmatskas) for more up-to-date news, articles and tips.

Share this post on

text=Scaffolding%20DbContext%20and%20Models%20with%20EntityFramework%20Core%202.0%20and%20th dbcontext-and-mo
(https://www.facebook.com/sharer/sharer.php?u=https://cmatskas.com/scaffolding-dbcontext-and-models-with-entityframework-core-2-0-and-the-cli/)
(http://www.linkedin.com/shareArticle?mini=true&url=https://cmatskas.com/scaffolding-dbcontext-and-m

cli/&title=Scaffolding%20DbContext%20and%20Models%20with%20EntityFramework%20Core%202.0%20and%
(https://plus.google.com/share?url=https://cmatskas.com/scaffolding-dbcontext-and-models-with-entityframework-core-2-0-and-the-cli/)

Getting started with public speaking, a guide for technical people (/getting-started-with-public-speaking-a-guide-for-technical-people/)

Blog Home (https://cmatskas.com)

Working with Application Insights and NLog in Console apps (.NET) (/working-with-application-insights-and-nlog-in-console-apps-net/)

11 Comments **cmatskas** **Login**

Recommend 1 **Share** **Sort by Best**

Join the discussion…

LOG IN WITH **OR SIGN UP WITH DISQUS** ?

Name

toba` • a year ago
I don't post comments often, but this is how tutorial should look. Simple, effective, concise. I was doing the same process like you wrote here every time, but the problem was that ERLANG_HOME wasn't being set after installation. So you really helped me. Great job and thanks!
2 • Reply • Share ›

ErikEJ • 2 months ago
You can also use SQLite Toolbox for an even simpler experience, that does not pollute your solution with desgin time dlls
1 • Reply • Share ›

Python Bouy • a year ago
this blog helped me to setup RabbitMQ, its really nice explained had some error but I set these path RABBITMQ_BASE C:\Users\UserName\AppData\RabbitMQ;
RABBITMQ_CONFIG_FILE == C:\Users\username\AppData\RabbitMQ and all set.
1 • Reply • Share ›

Muhammad Tri Wibowo • 9 days ago
in commands for add package u need to give single space for -v
dotnet add package Microsoft.EntityFrameworkCore.Tools.DotNet-v 2.0.0
should be :
dotnet add package Microsoft.EntityFrameworkCore.Tools.DotNet -v 2.0.0

• Reply • Share ›

cmsganesh cmsgp • 4 months ago
Useful...thanks
• Reply • Share ›

Mayron Guevara • 5 months ago
All that i was finding.... excellent post
• Reply • Share ›

Narayanan Ts • 9 months ago
Great Post! Explained installation in a clear way! Thanks!
• Reply • Share ›

mher • a year ago
Thank you, This serves me very much.
• Reply • Share ›

Pavel Vlasov • 2 years ago
Thank you much! That "messaging that just works" definitely requires such a helpful tutorial :-D Personally I met an error when the service won't start - solved it reinstalling Erlang from x86 to x64 + checked the option to install Microsoft dll's. Also recent releases does not accept guest/guest creds - google for "Cannot login in RabbitMQ Management web console" to solve that.
• Reply • Share ›

Christos Matskas Mod • 3 years ago
Hi **@xaralabos karypidis**, many thanks for the comment. I look forward to attending and I will do another post with my impressions.
• Reply • Share ›

xaralabos karypidis • 3 years ago
Wish you to enjoy it to the max. You will certainly have a great time!
• Reply • Share ›

ALSO ON **CMATSKAS**

Deploying Functions with ARM templates and inline code
2 comments • 10 months ago
AvatarChristos Matskas — I'm afraid that no, this wouldn't work. This solution for very simple scenarios and not one I would recommend. With Functions moving to precompiled code (not .csx) means that you won't be able to use this …

Automate login for Azure Powershell scripts
3 comments • a year ago
AvatarChristos Matskas — Hi Jonathan_Harding , this is correct. This is a service account that can be used in the context of Azure to execute scripts under a limited-permissions account

Introduction Azure Functions Deployment Slots
6 comments • 6 months ago
AvatarChristos Matskas — I'm afraid that this is not an option. The whole of the Azure Functions code is deployed as whole, so per-Function slot swap is not available.

Azure Functions, Node.js and Environment Variables
1 comment • 4 months ago
Avatarjustintimecoder — Nice article, there are way too many tutorials on functions that just show the quick fix portal solution(makes my skin crawl). It's good to see one with a devflow that could be used professionally.

Subscribe **Add Disqus to your siteAdd DisqusAdd** **Privacy**