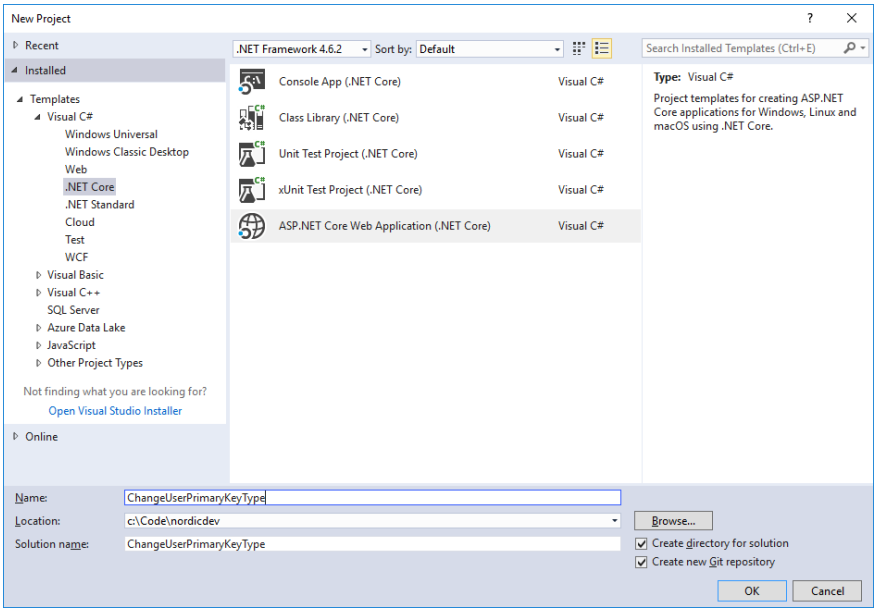
Alec Papierniak Follow
Co-founder at Nordic Dev
Apr 25, 2017 · 6 min read

Changing the primary key type of ApplicationUser in ASP.NET Identity Core

ASP.NET Identity Core is a great collection of code. Evolved from ASP.NET Membership, ASP.NET Identity Core is the latest and greatest solution from Microsoft for providing user management. By offloading the heavy lifting of common user management tasks to a popular open source library, developers can save time and reduce bugs. If you're here, you're likely already using Identity in a project of your own.

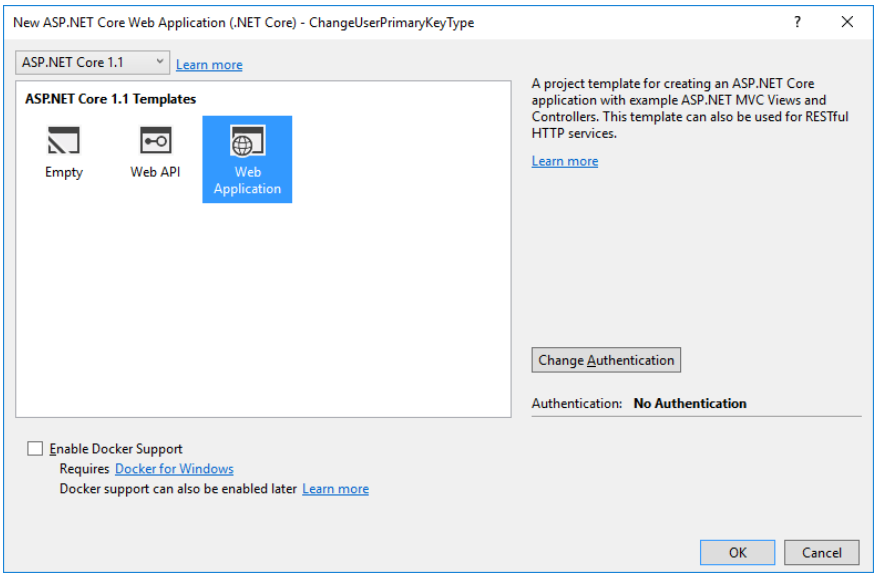
The default options when you add Identity Core to your project fit a wide variety of use cases. It is common, however, to want to change the type of the primary key used for the default user class. Out of the box, this type is string. Your requirements may specify a different type. Let's take a look and see how we can update the type to fit our needs. We're going to switch from the default type string, to int.

First, let's create a new ASP.NET Core Web project. I'm using Visual Studio 2017 in this example, but the process is very similar in 2015. I'm calling mine ChangeUserPrimaryKeyType. The code is available on Github over here

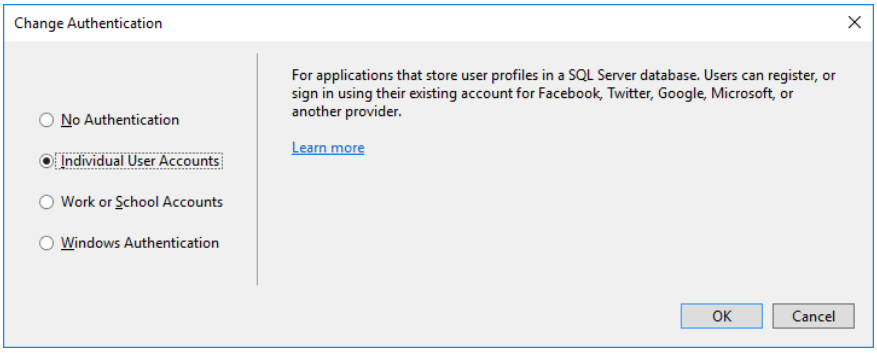


Click OK.

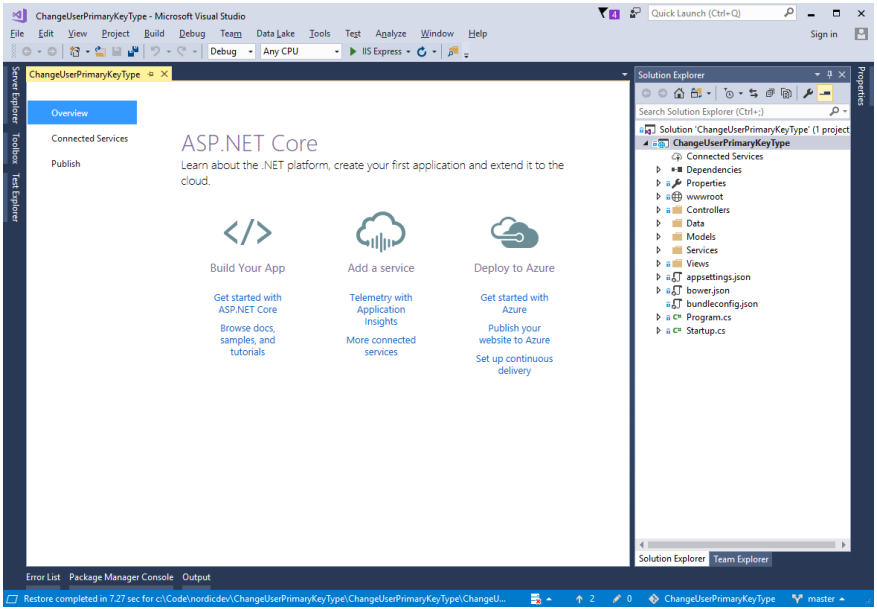
On the next screen, select Web Application



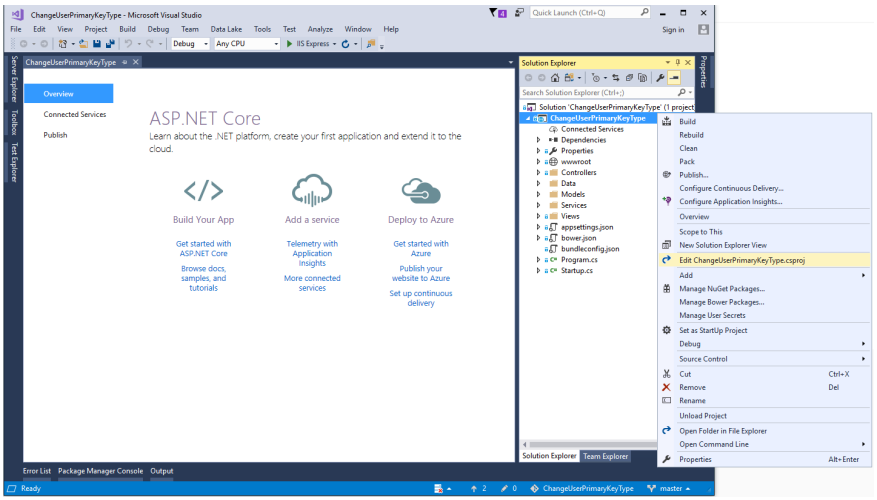
The default option is to have no authentication. We're going to change this to Individual User Accounts. This will tell Visual Studio to do all the heavy lifting of including ASP.NET Identity Core, EntityFrameworkCore, and a few other supporting packages. Click the Change Authentication button, and select Individual User Accounts

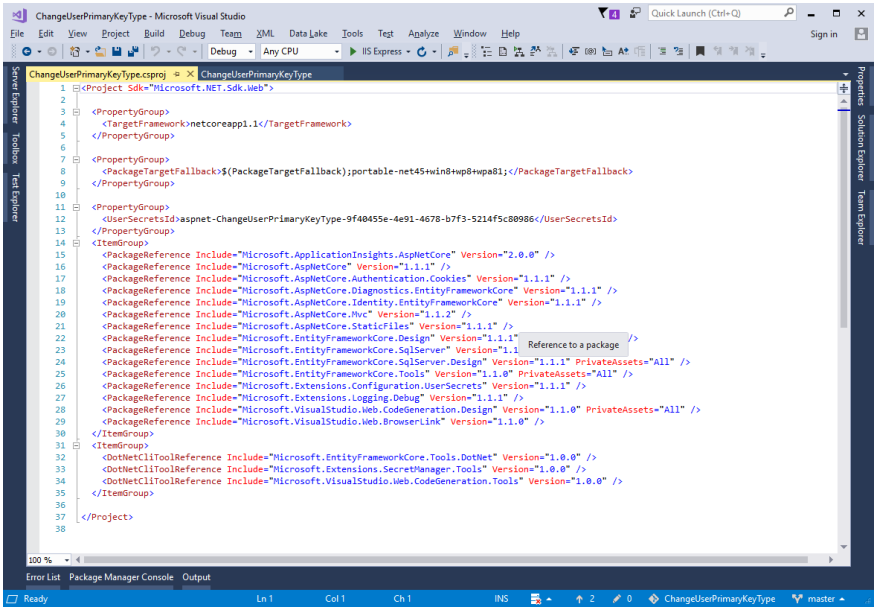


Click OK, and OK once again. Visual Studio will scaffold out the application, and in a few seconds you'll be greeted with a screen similar to this

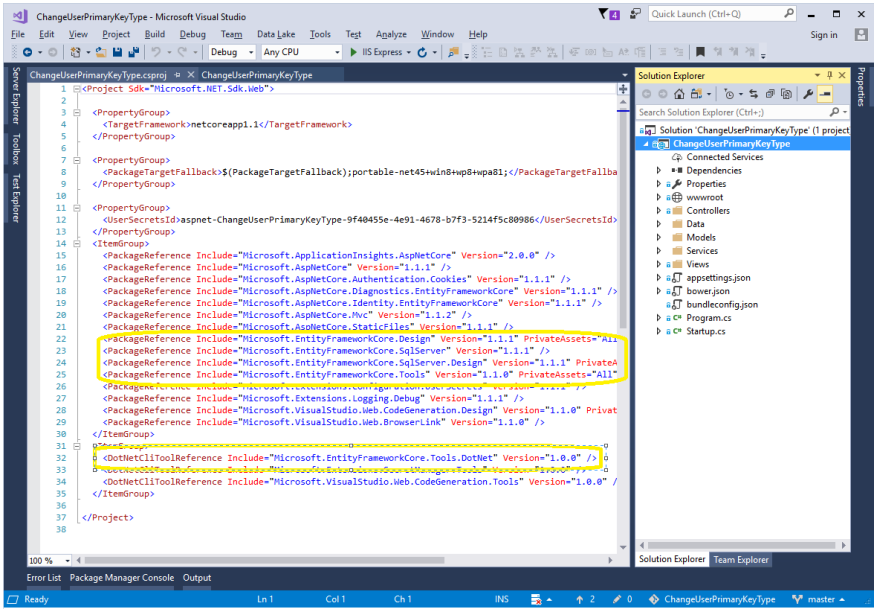


Let's take a moment and look at all the packages that were included by default. Since I'm using VS2017, I'll take a look at the csproj file. If you're using VS2015, take a look at the project.json file. Keep in mind that in VS2017 Microsoft switched from the project.json back to a csproj file. Right click on Edit ChangeUserPrimaryKeyType.csproj





By default, the application will be configured to use EntityFrameworkCore—you can see that a few EFCore packages were brought in automatically

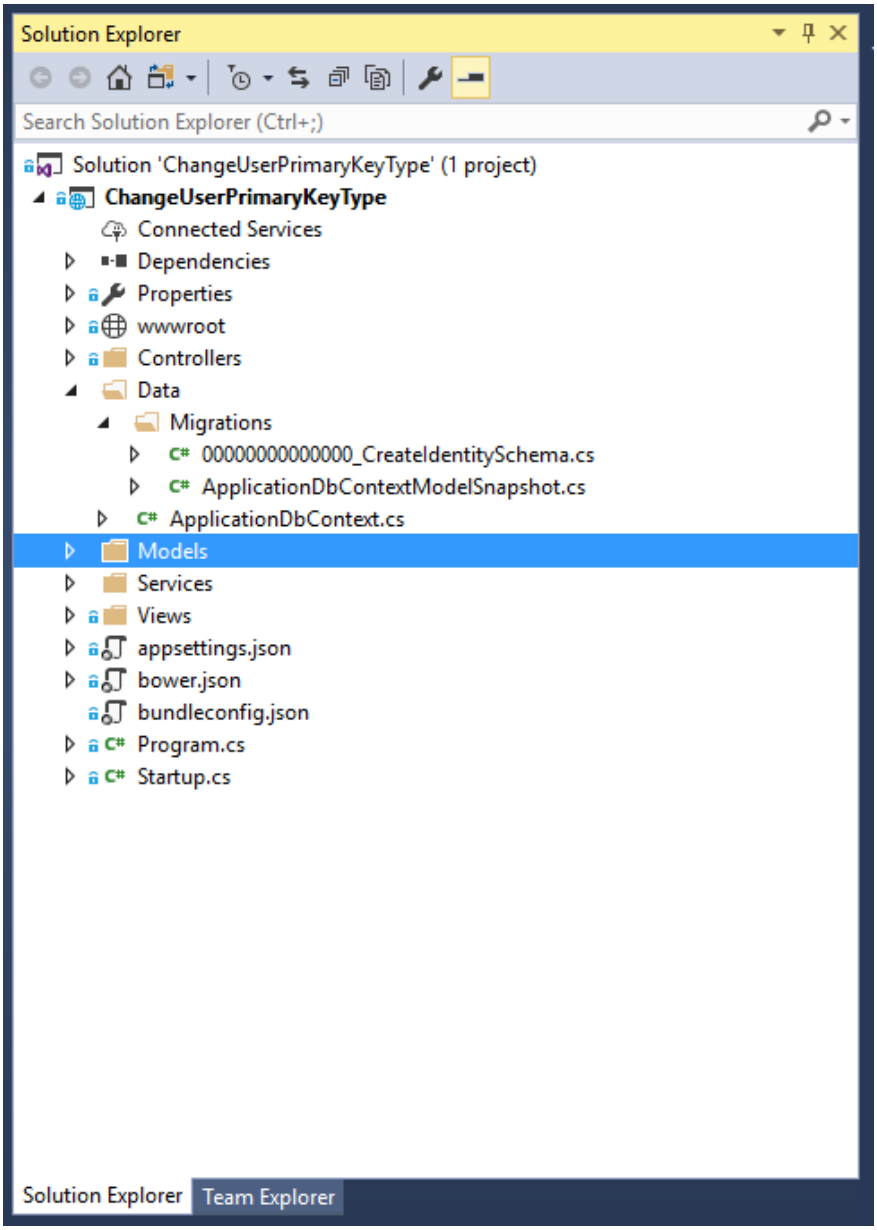


Microsoft.EntityFrameworkCore.Tools provides the tools we need to use the Package Manage Console to add migrations, perform migrations, and all sorts of other fun things. Microsoft.EntityFrameworkCore.Tools.DotNet provides similar functionality, but from the command line.

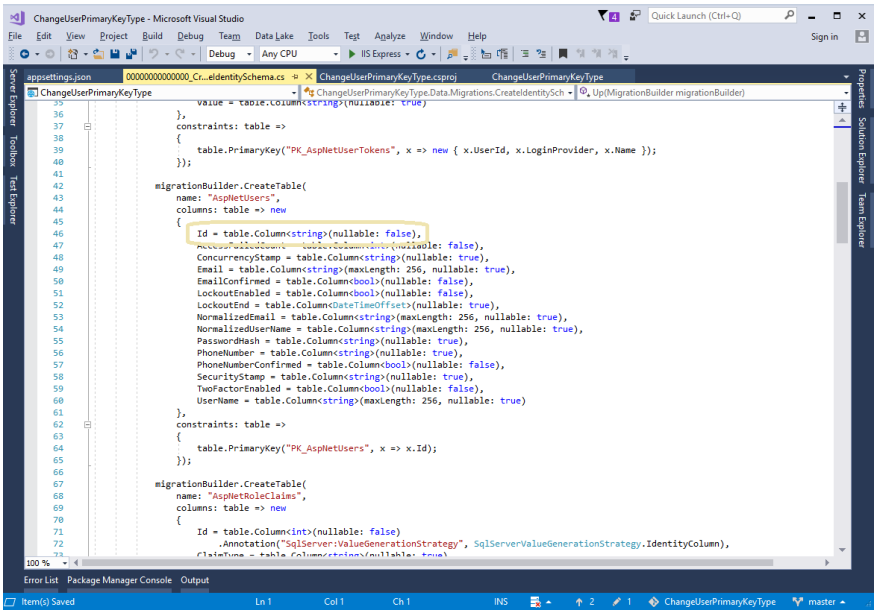
Speaking of the command line, an extremely helpful called Open Command Line from Mads Kristensen. If you don't already have it installed, I highly recommend it.

In addition to the EntityFrameworkCore packages, you'll also see Microsoft.AspNetCore.Identity.EntityFrameworkCore, Microsoft.AspNetCore.Authentication.Cookies. These two packages are brought in when we selected Individual User Accounts when we created the project.

Before we get started, let's take a quick look at the Data->Migrations folder

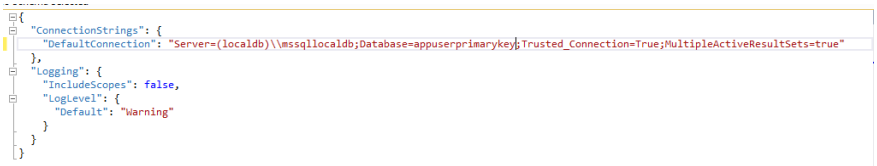


There's already a migration in there. This migration will, surprisingly, create the schema needed to support Identity. Open up that migration and let's take a look at the AspNetUsers table definition contained within.



You can see here this migration will create a column named Id, of type string.

Let's take a look at the appsettings.json file and see the connection string.



I've changed the database from the default to 'appuserprimarykey'.

Let's move on with changing the key type. Open up the ApplicationUser.cs file

Change

```
public class ApplicationUser : IdentityUser
```

to

```
public class ApplicationUser : IdentityUser<int>
```

and Save.

Next, open up Startup.cs. Under ConfigureServices, you'll see the block of code where VS configured the identity service

```
services.AddIdentity<ApplicationUser, IdentityRole>()
.AddEntityFrameworkStores<ApplicationDbContext>()
.AddDefaultTokenProviders();
```

Since we want to use an int as the primary key type, we need to make a few changes here as well. We need to tell IdentityRole which type to use (int), and we need to pass the same type to AddEntityFrameworkStores, like this

```
services.AddIdentity<ApplicationUser, IdentityRole<int>>()
.AddEntityFrameworkStores<ApplicationDbContext, int>()
.AddDefaultTokenProviders();
```

We need to update one more file. Open up ApplicationDbContext.cs. The default class definition should look like this

```
public class ApplicationDbContext :
IdentityDbContext<ApplicationUser>
```

Since we want to use int as our key type, and we need to update IdentityRole to do this as well, here's what the new definition should look like

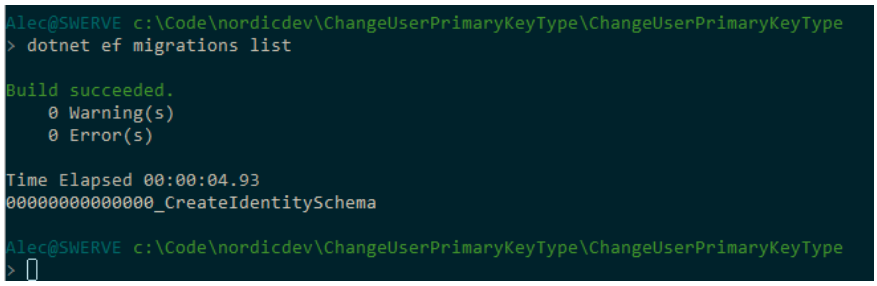
```
public class ApplicationDbContext :
IdentityDbContext<ApplicationUser, IdentityRole<int>, int>
```

Your project should now build, hooray!

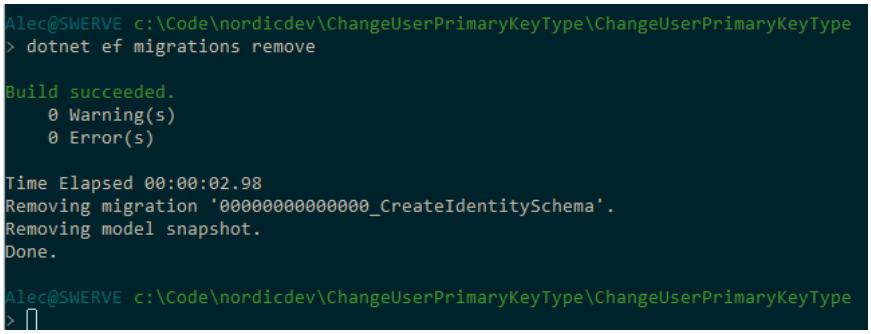
Up to this point, we haven't created our database. Since the default migration was created when we were going to use a string as the key type, we need to remove that migration, add a new migration, and then apply the new migration.

I'm going to use the command line to do these tasks. If you installed the Open Command Line extension, right click on your project name and select Open Command Line. If not, open a command line and cd to your project directory.

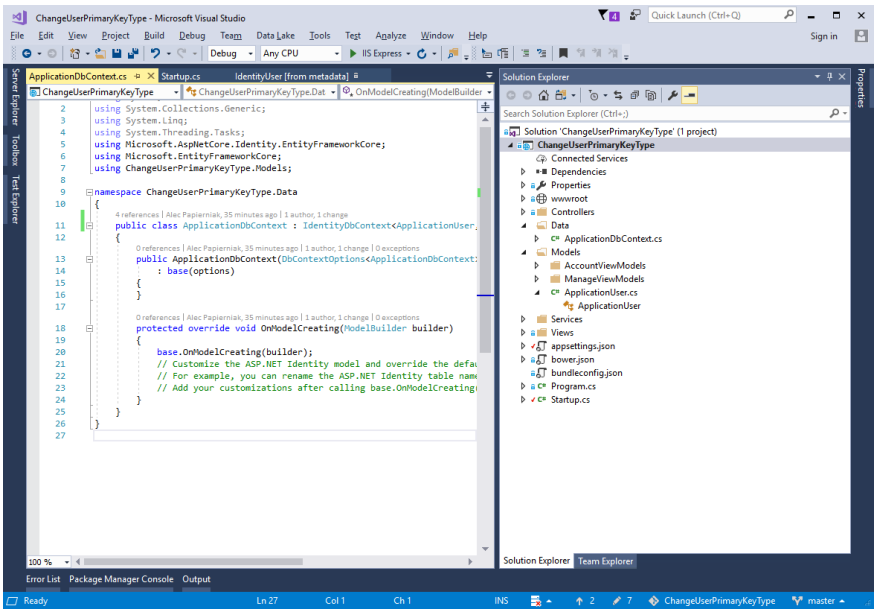
First, let's list the migrations. Since we only have one dbcontext, we don't need to specify it when calling dotnet ef on the command line



Great, just the default schema. Let's remove that and keep going. Since there's only one migration, we don't need to specify it when calling migrations remove



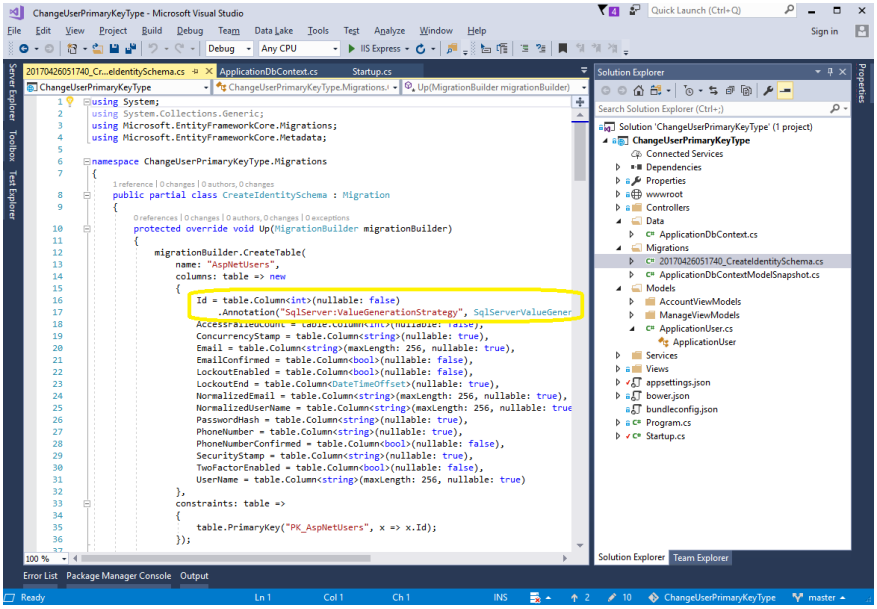
Check back in VS and verify that the migration has been removed from the Data->Migrations folder



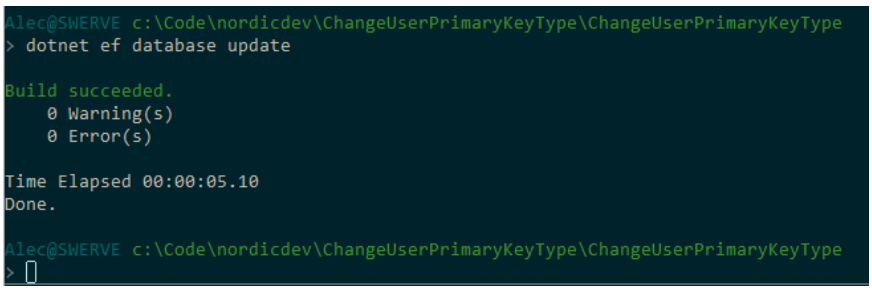
Alright, the default migration was removed. Time to create a new migration. Back to the command line!

```
Alec@SWERVE c:\Code\nordicdev\ChangeUserPrimaryKeyType\ChangeUserPrimaryKeyType
> dotnet ef migrations add "CreateIdentitySchema"

Build succeeded.
    0 Warning(s)
    0 Error(s)

Time Elapsed 00:00:03.12
Done. To undo this action, use 'ef migrations remove'

Alec@SWERVE c:\Code\nordicdev\ChangeUserPrimaryKeyType\ChangeUserPrimaryKeyType
>
```

Notice, we didn't specify a migrations folder, so by default the tool will create a folder called Migrations at the root level of the project. Back in Visual Studio, open up the newly created migration folder and take a look at the Id column definition of the AspNetUsers table

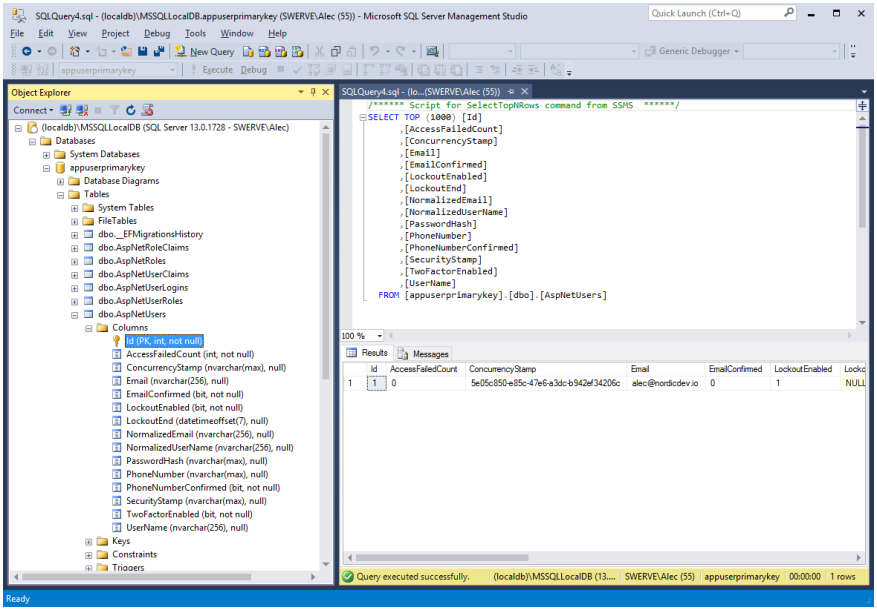


Hooray—what once was string, now is int. Time to go back to the command line and apply the migration



Fire up the application. Click the Register link, and create a new account.

To verify, I'm going to use SSMS to connect to the same database as specified in my appsettings.json file. Take a look at the AspNetUsers table, even select the top 1000 rows if you'd like



Hooray! We now have a sparkling new ASP.NET Core web application, using ASP.NET Identity Core, and we modified the user definition to have a primary key of type int. Fun stuff!