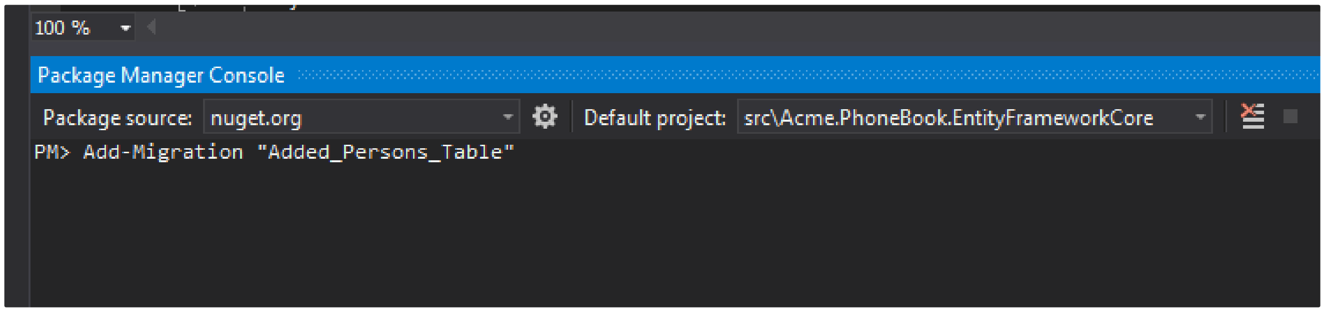
Edit

Database Migrations for Person

We use **EntityFramework Code-First migrations** to migrate database schema. Since we added **Person entity**, our DbContext model is changed. So, we should create a **new migration** to create the new table in the database.

Open **Package Manager Console**, run the **Add-Migration "Added_Persons_Table"** command as shown below:



Package source: nuget.org
Default project: src\Acme.PhoneBook.EntityFrameworkCore

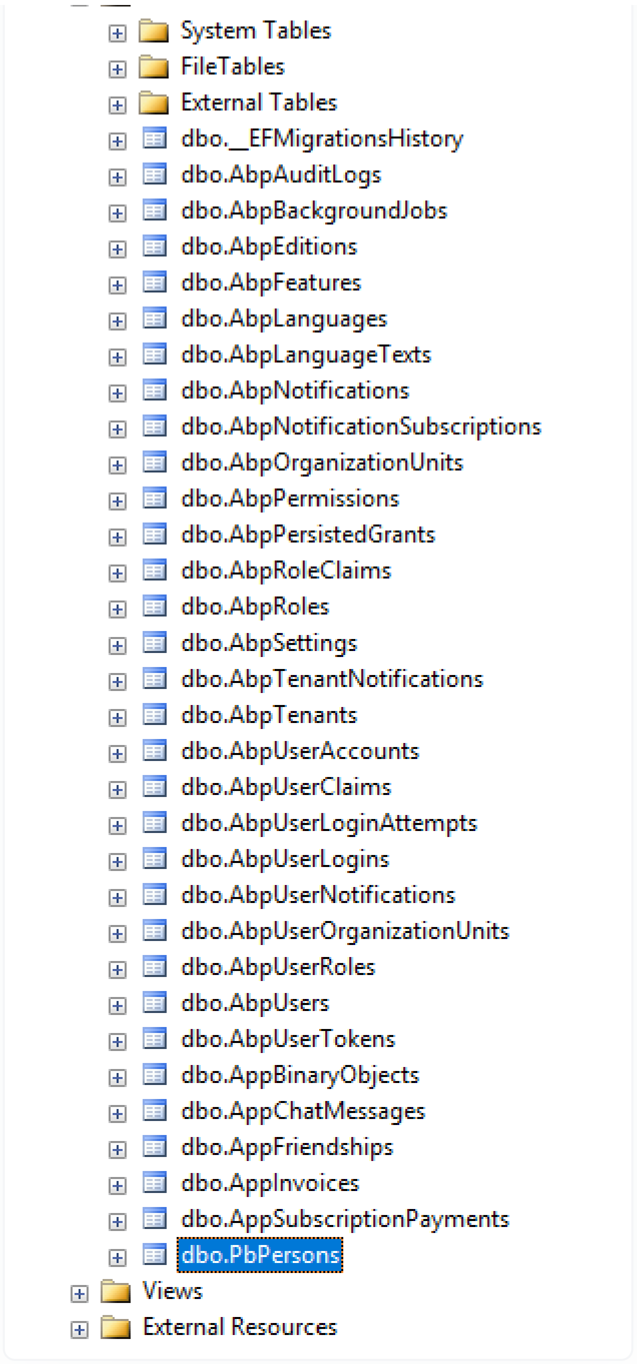
```
PM> Add-Migration "Added_Persons_Table"
```

Feedback

This command will add a **migration class** named "**Added_Persons_Table**" as shown below:

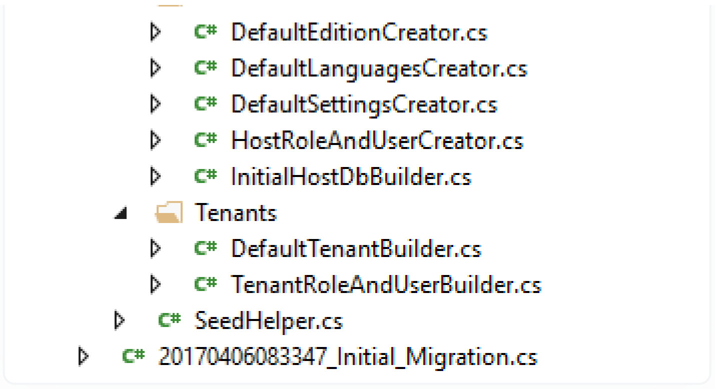
```
public partial class Added_Persons_Table : Migration
{
    protected override void Up(MigrationBuilder migrationBuilder)
    {
        migrationBuilder.CreateTable(
            name: "PbPersons",
            columns: table => new
            {
                Id = table.Column(nullable: false)
                    .Annotation("SqlServer:ValueGenerationStrategy", SqlServerValue
                CreationTime = table.Column(nullable: false),
                CreatorUserId = table.Column(nullable: true),
                DeleterUserId = table.Column(nullable: true),
                DeletionTime = table.Column(nullable: true),
                EmailAddress = table.Column(maxLength: 255, nullable: true),
                IsDeleted = table.Column(nullable: false),
                LastModificationTime = table.Column(nullable: true),
            },
            constraints: table =>
            {
                table.PrimaryKey("PK_PbPersons", x => x.Id);
            });
    }

    protected override void Down(MigrationBuilder migrationBuilder)
    {
        migrationBuilder.DropTable(
            name: "PbPersons");
    }
}
```

We don't have to know so much about format and rules of this file. But, it's suggested to have a basic understanding of migrations. In the same Package Manager Console, we write **Update-Database** command in order to apply the new migration to database. After updating, we can see that **PbPersons table** is added to database.



```
dbo.__EFMigrationsHistory
dbo.AbpAuditLogs
dbo.AbpBackgroundJobs
dbo.AbpEditions
dbo.AbpFeatures
dbo.AbpLanguages
dbo.AbpLanguageTexts
dbo.AbpNotifications
dbo.AbpNotificationSubscriptions
dbo.AbpOrganizationUnits
dbo.AbpPermissions
dbo.AbpPersistedGrants
dbo.AbpRoleClaims
dbo.AbpRoles
dbo.AbpSettings
dbo.AbpTenantNotifications
dbo.AbpTenants
dbo.AbpUserAccounts
dbo.AbpUserClaims
dbo.AbpUserLoginAttempts
dbo.AbpUserLogins
dbo.AbpUserNotifications
dbo.AbpUserOrganizationUnits
dbo.AbpUserRoles
dbo.AbpUsers
dbo.AbpUserTokens
dbo.AppBinaryObjects
dbo.AppChatMessages
dbo.AppFriendships
dbo.AppInvoices
dbo.AppSubscriptionPayments
dbo.PbPersons
```

But this new table is empty. In ASP.NET Zero, there are some classes to fill initial data for users and settings:


DefaultEditionCreator.cs
DefaultLanguagesCreator.cs
DefaultSettingsCreator.cs
HostRoleAndUserCreator.cs
InitialHostDbBuilder.cs
Tenants
DefaultTenantBuilder.cs
TenantRoleAndUserBuilder.cs
SeedHelper.cs
20170406083347_Initial_Migration.cs

So, we can add a separated class to fill some people to database as shown below:

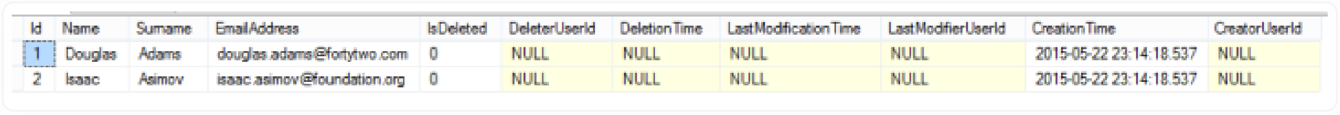
```
namespace Acme.PhoneBookDemo.Migrations.Seed.Host
{
    public class InitialPeopleCreator
    {
        private readonly PhoneBookDemoDbContext _context;

        public InitialPeopleCreator(PhoneBookDemoDbContext context)
        {
            _context = context;
        }

        public void Create()
        {
            var douglas = _context.Persons.FirstOrDefault(p => p.EmailAddress == "d
            if (douglas == null)
            {
                _context.Persons.Add(
                    new Person
                    {
                        Name = "Douglas",
                        Surname = "Adams",
                        EmailAddress = "douglas.adams@fortytwo.com"
                    });
            }

            var asimov = _context.Persons.FirstOrDefault(p => p.EmailAddress == "is
            if (asimov == null)
            {
                _context.Persons.Add(
                    new Person
```

```
                        EmailAddress = "isaac.asimov@foundation.org"
                    });
                }
            }
        }
    }
```

These type of default data is good since we can also use these data in **unit tests**. Surely, we should be careful about seed data since this code will always be executed in each **PostInitialize** of your PhoneBookEntityFrameworkCoreModule. This class (InitialPeopleCreator) is created and called in **InitialHostDbBuilder** class. This is not so important, just for a good code organization (see source codes).

```
public class InitialHostDbBuilder
{
    //existing codes...

    public void Create()
    {
        //existing code...
        new InitialPeopleCreator(_context).Create();

        _context.SaveChanges();
    }
}
```

We run our project again, it runs seed and adds two people to PbPersons table:



Id	Name	Surname	EmailAddress	IsDeleted	DeleterUserId	DeletionTime	LastModificationTime	LastModifierUserId	CreationTime	CreatorUserId
1	Douglas	Adams	douglas.adams@fortytwo.com	0	NULL	NULL	NULL	NULL	2015-05-22 23:14:18.537	NULL
2	Isaac	Asimov	isaac.asimov@foundation.org	0	NULL	NULL	NULL	NULL	2015-05-22 23:14:18.537	NULL

Next

- Creating Unit Tests for Person Application Service