

Getting Started with EF Core on ASP.NET Core with a New database

📅 04/07/2017 • ⌚ 4 minutes to read • Contributors  [all](#)

In this article

[Create a new project in Visual Studio 2017](#)

[Install Entity Framework Core](#)

[Create the model](#)

[Register your context with dependency injection](#)

[Create your database](#)

[Create a controller](#)

[Run the application](#)

[Additional Resources](#)

In this walkthrough, you will build an ASP.NET Core MVC application that performs basic data access using Entity Framework Core. You will use migrations to create the database from your EF Core model. See [Additional Resources](#) for more Entity Framework Core tutorials.

This tutorial requires:

- [Visual Studio 2017 15.3](#) with these workloads:
 - **ASP.NET and web development** (under **Web & Cloud**)
 - **.NET Core cross-platform development** (under **Other Toolsets**)
- [.NET Core 2.0 SDK](#).

💡 Tip

You can view this article's [sample](#) on GitHub.

Create a new project in Visual Studio 2017

- **File > New > Project**
- From the left menu select **Installed > Templates > Visual C# > .NET Core**.
- Select **ASP.NET Core Web Application**.
- Enter **EFGetStarted.AspNetCore.NewDb** for the name and click **OK**.
- In the **New ASP.NET Core Web Application** dialog:
 - Ensure the options **.NET Core** and **ASP.NET Core 2.0** are selected in the drop down lists
 - Select the **Web Application (Model-View-Controller)** project template
 - Ensure that **Authentication** is set to **No Authentication**

- Click **OK**

Warning: If you use **Individual User Accounts** instead of **None** for **Authentication** then an Entity Framework Core model will be added to your project in `Models\IdentityModel.cs`. Using the techniques you will learn in this walkthrough, you can choose to add a second model, or extend this existing model to contain your entity classes.

Install Entity Framework Core

Install the package for the EF Core database provider(s) you want to target. This walkthrough uses SQL Server. For a list of available providers see [Database Providers](#).

- **Tools > NuGet Package Manager > Package Manager Console**

- Run `Install-Package Microsoft.EntityFrameworkCore.SqlServer`

We will be using some Entity Framework Core Tools to create a database from your EF Core model. So we will install the tools package as well:

- Run `Install-Package Microsoft.EntityFrameworkCore.Tools`

We will be using some ASP.NET Core Scaffolding tools to create controllers and views later on. So we will install this design package as well:

- Run `Install-Package Microsoft.VisualStudio.Web.CodeGeneration.Design`

Create the model

Define a context and entity classes that make up the model:

- Right-click on the **Models** folder and select **Add > Class**.
- Enter **Model.cs** as the name and click **OK**.
- Replace the contents of the file with the following code:

C#

 Copy

```
using Microsoft.EntityFrameworkCore;
using System.Collections.Generic;

namespace EFGetStarted.AspNetCore.NewDb.Models
{
    public class BloggingContext : DbContext
    {
        public BloggingContext(DbContextOptions<BloggingContext> options)
```

```
        : base(options)
    { }

    public DbSet<Blog> Blogs { get; set; }
    public DbSet<Post> Posts { get; set; }
}

public class Blog
{
    public int BlogId { get; set; }
    public string Url { get; set; }

    public List<Post> Posts { get; set; }
}

public class Post
{
    public int PostId { get; set; }
    public string Title { get; set; }
    public string Content { get; set; }

    public int BlogId { get; set; }
    public Blog Blog { get; set; }
}
}
```

Note: In a real app you would typically put each class from your model in a separate file. For the sake of simplicity, we are putting all the classes in one file for this tutorial.

Register your context with dependency injection

Services (such as `BloggingContext`) are registered with [dependency injection](#) during application startup. Components that require these services (such as your MVC controllers) are then provided these services via constructor parameters or properties.

In order for our MVC controllers to make use of `BloggingContext` we will register it as a service.

- Open **Startup.cs**
- Add the following `using` statements:

C#

 Copy

```
using EFGetStarted.AspNetCore.NewDb.Models;
using Microsoft.EntityFrameworkCore;
```

Add the `AddDbContext` method to register it as a service:

- Add the following code to the `ConfigureServices` method:

```
C# Copy

// This method gets called by the runtime. Use this method to add services to the
public void ConfigureServices(IServiceCollection services)
{
    services.AddMvc();

    var connection = @"Server=(localdb)\mssqllocaldb;Database=EFGetStarted.AspNetC
    services.AddDbContext<BloggContext>(options => options.UseSqlServer(connect
}
```

Note: A real app would generally put the connection string in a configuration file. For the sake of simplicity, we are defining it in code. See [Connection Strings](#) for more information.

Create your database

Once you have a model, you can use [migrations](#) to create a database.

- Open the PMC:

Tools → NuGet Package Manager → Package Manager Console

- Run `Add-Migration InitialCreate` to scaffold a migration to create the initial set of tables for your model. If you receive an error stating `The term 'add-migration' is not recognized as the name of a cmdlet`, close and reopen Visual Studio.
- Run `Update-Database` to apply the new migration to the database. This command creates the database before applying migrations.

Create a controller

Enable scaffolding in the project:

- Right-click on the **Controllers** folder in **Solution Explorer** and select **Add > Controller**.
- Select **Minimal Dependencies** and click **Add**.
- You can ignore or delete the *ScaffoldingReadMe.txt* file.

Now that scaffolding is enabled, we can scaffold a controller for the `Blog` entity.

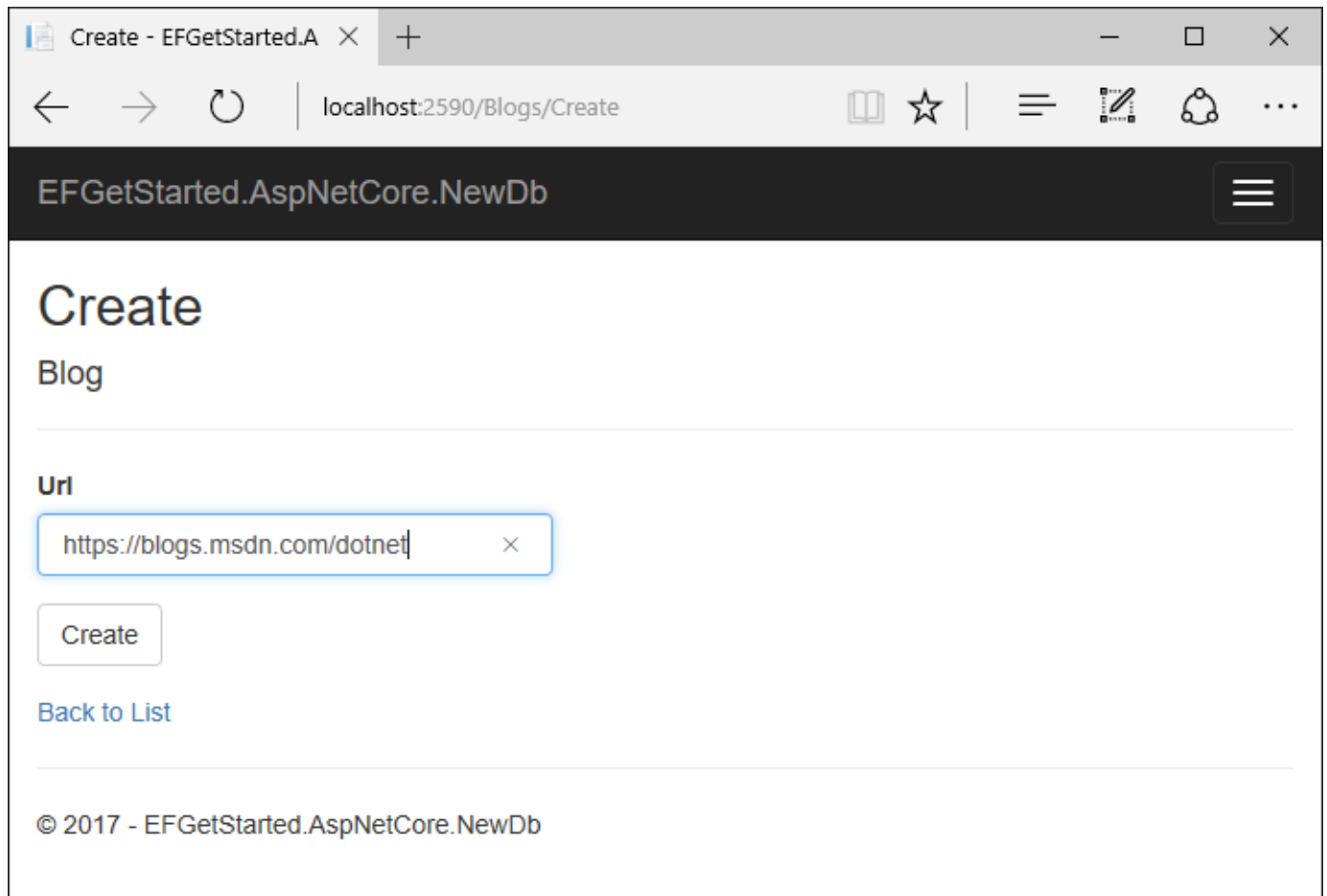
- Right-click on the **Controllers** folder in **Solution Explorer** and select **Add > Controller**.
- Select **MVC Controller with views, using Entity Framework** and click **Ok**.

- Set **Model class** to **Blog** and **Data context class** to **BloggingContext**.
- Click **Add**.

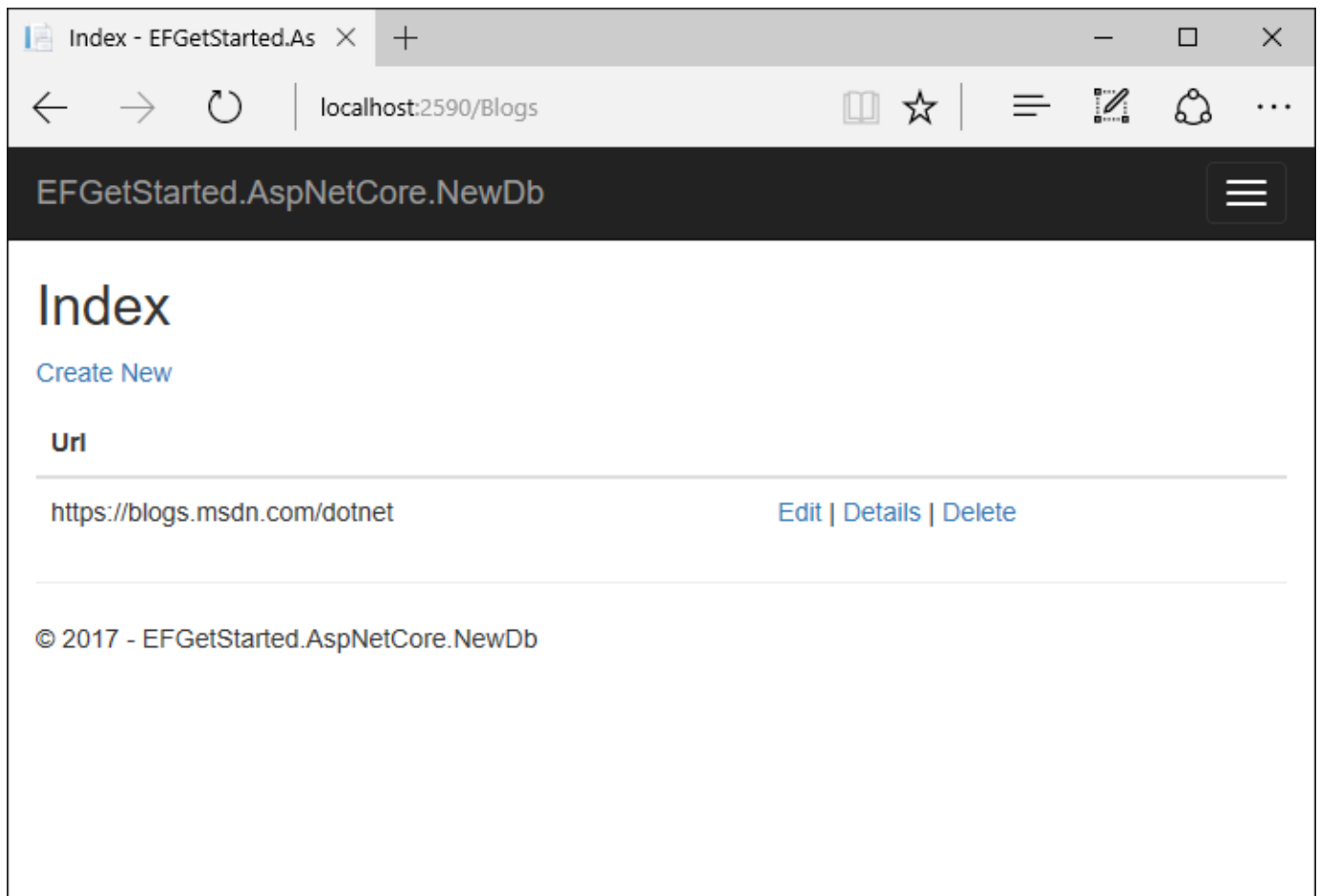
Run the application

Press F5 to run and test the app.

- Navigate to
- Use the create link to create some blog entries. Test the details and delete links.



The screenshot shows a web browser window with the title 'Create - EFGetStarted.A'. The address bar shows 'localhost:2590/Blogs/Create'. The page has a dark header with the text 'EFGetStarted.AspNetCore.NewDb' and a hamburger menu icon. The main content area is titled 'Create Blog'. Below the title, there is a form with a label 'Url' and a text input field containing 'https://blogs.msdn.com/dotnet'. Below the input field is a 'Create' button. At the bottom of the form, there is a link 'Back to List'. The footer of the page shows '© 2017 - EFGetStarted.AspNetCore.NewDb'.



Additional Resources

- [EF - New database with SQLite](#) - a cross-platform console EF tutorial.
- [Introduction to ASP.NET Core MVC on Mac or Linux](#)
- [Introduction to ASP.NET Core MVC with Visual Studio](#)
- [Getting started with ASP.NET Core and Entity Framework Core using Visual Studio](#)

Note

The feedback system for this content will be changing soon. Old comments will not be carried over. If content within a comment thread is important to you, please save a copy. For more information on the upcoming change, [we invite you to read our blog post](#).