

数据访问技术入门

2019年3月26日 22:36

重要概念

Entity Framework (EF) Core 是轻量化、可扩展、[开源](#)和跨平台的数据访问技术，它还是一种对象关系映射器 (ORM)，它使 .NET 开发人员能够使用面向对象的思想处理数据库，它消除了开发人员通常需要编写大量数据访问代码的需要。

EF Core 支持多个数据库引擎，请参阅[数据库提供程序](#)了解详细信息。

系统必备

- EF Core 是一个 .NET Standard 2.0 库，因此它能够在其它平台使用。
- 安装 .NET Core SDK 即可使用。
- EF Core 可以在 Xamarin 和 .NET Native 等其他 .NET 实现上运行。
- 不同数据库需要 EF Core 数据库提供程序支持。

安装相关包

Microsoft.EntityFrameworkCore
Microsoft.EntityFrameworkCore.Design
Microsoft.EntityFrameworkCore.Tools

由微软维护的支持程序

Microsoft.EntityFrameworkCore.SqlServer
Microsoft.EntityFrameworkCore.Sqlite
Microsoft.EntityFrameworkCore.InMemory

用命令行创建数据库

```
dotnet new console -o ConsoleAppSample
cd ConsoleAppSample
dotnet add package Microsoft.EntityFrameworkCore.Sqlite
dotnet add package Microsoft.EntityFrameworkCore.Design
dotnet restore
```

```

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

namespace ConsoleAppSample
{
    public class BloggingContext : DbContext
    {
        public DbSet<Blog> Blogs { get; set; }
        public DbSet<Post> Posts { get; set; }

        protected override void
OnConfiguring(DbContextOptionsBuilder optionsBuilder)
        {
            optionsBuilder.UseSqlite("Data
Source=blogging.db");
        }

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

            public ICollection<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; }
        }
    }
}

```

dotnet ef migrations add InitialCreate

dotnet ef database update

```
using System;
```

```

namespace ConsoleAppSample
{
    public class Program
    {

```

```

        public static void Main()
        {
            using (var db = new BloggingContext())
            {
                db.Blogs.Add(new Blog { Url =
"www.xcode.me" });

                var count = db.SaveChanges();
                Console.WriteLine("{0} records saved to
database", count);

                Console.WriteLine();
                Console.WriteLine("All blogs in
database:");

                foreach (var blog in db.Blogs)
                {
                    Console.WriteLine(" - {0}",
blog.Url);
                }
            }
        }
    }
}

```

dotnet ef migrations add
dotnet ef database update

安装支持包

在 visual Studio 创建一个能够访问数据库的 ASP.NET Core 应用。

Install-Package Microsoft.EntityFrameworkCore.Sqlite

Microsoft.AspNetCore.App 元包

来自 <<https://www.nuget.org/packages/Microsoft.AspNetCore.App/2.2.2>>

创建模型

```

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 ICollection<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; }
    }
}

```

使用依赖注入注册上下文

```

var services.AddDbContext<BloggingContext>(options =>
options.UseSqlServer(connection));

```

通过迁移工具创建数据库

```

Add-Migration InitialCreate
Update-Database

```

创建控制器

创建具有增删改查的 BlogsController.cs 控制器便于测试。

SQLiteStudio

来自 <<https://github.com/pawelsalawa/sqlitestudio>>

