عنوان مقاله: DbContext چیست؟ + بهترین روش استفاده در EF Core
نویسنده مقاله: تیم فنی نیک‌آموز
تاریخ انتشار: ۲۹ بهمن ۱۴۰۲
منبع: https://nikamooz.com/what-is-dbcontext/

DbContext چیست ؟ اگر به‌تازگی وارد دنیای .NET شده باشید، احتمالاً تاکنون این پرسش در ذهنتان شکل گرفته است. DbContext نقش بسزایی در اپلیکیشن‌های Entity Framework Core ایفا کرده و مشابه یک پل ارتباطی میان پایگاه داده (Database) و اپلیکیشن عمل می‌کند. در این مقاله قصد داریم به بررسی چیستی DbContext ، مزایا و معایب، کاربردها و بهترین شیوه‌های استفاده از آن در EF Core بپردازیم.

منظور از DbContext چیست؟

در پاسخ به این سؤال که DbContext چیست ، یک جواب شفاف وجود دارد. DbContext همانند یک پل ارتباطی میان اپلیکیشن و پایگاه داده در Entity Framework Core عمل می‌کند. به‌واسطه استفاده از DbContext ، دسترسی به داده‌ها ساده‌سازی می‌شود و کارکردهایی مانند مدیریت اتصال‌ها (Connections)، رسیدگی به مجموعه موجودیت (Entity Set)، پیگیری و ذخیره‌سازی تغییرات تسهیل می‌یابند. بدین ترتیب، نگهداری کد (Code Maintenance) بهبود پیدا کرده و قابلیت‌های مربوط به کوئری‌نویسی به‌صورت کارآمد قابل انجام خواهند بود.

مزایای DbContext چیست ؟

مهم‌ترین مزایای DbContext در ادامه فهرست شده‌اند:

- **ساده‌سازی دسترسی به داده‌ها:** DbContext به‌دلیل ارائه یک رابط یکپارچه برای تعامل با پایگاه داده، شما را از به کار بردن کوئری‌های پیچیده SQL بی‌نیاز می‌کند.
- **بهبود نگهداری کد:** منطق دسترسی به داده‌ها در درون DbContext کپسوله سازی (Encapsulation) می‌شوند و به‌دنبال آن، قابلیت استفاده مجدد (Reusability) کد تقویت خواهد شد.
- **کوئری‌نویسی کارآمد:** توسعه‌دهنده می‌تواند به‌واسطه سینتکس LINQ، داده‌ها را استخراج کند و بدین روش، عملکرد بهتری از خود به نمایش بگذارد.
- **امکان پیگیری سریع تغییرات:** تغییرات اعمال‌شده روی موجودیت‌ها (Entities) به‌صورت خودکار قابل پیگیری هستند. در چنین شرایطی، پیش از ذخیره‌سازی، خیالتان از بابت سازگاری داده‌ها راحت خواهد بود.

معایب DbContext چیست ؟

نقاط ضعف DbContext عبارتند از:

- سربار مربوط به اضافه‌شدن یک لایه انتزاعی
- دسترسی محدود و کنترل کمتر روی کوئری‌های تولیدشده
- عدم وجود قابلیت Thread Safety
- وابستگی بالا به EF Core

تا این بخش از مقاله، به این پرسش پاسخ داده شد که DbContext چیست و چه مزایا و معایبی دارد. حال در ادامه قصد داریم به بررسی کاربردها عمده آن بپردازیم.

کاربرد های DbContext

اصلی‌ترین موارد استفاده DBContext به شرح زیر است:

- **عملیات CRUD:** می‌توان عملیات Create ، Read، Update و Delete را روی داده‌های درون جداول پایگاه داده اجرا کرد.
- **استخراج داده‌ها:** امکان فیلترسازی و واکشی زیرمجموعه‌ای از داده‌ها براساس معیارهای خاصی ازطریق کوئری‌های LINQ فراهم شده است.
- **ساخت اپلیکیشن داده‌محور:** در سناریوهایی همچون سیستم مدیریت محتوا، سیستم‌های تجارت الکترونیک (E-commerce) یا سایر موارد، امکان مدیریت تعاملات داده‌محور به‌صورت کارآمد وجود دارد.
- **یکپارچه‌سازی با سایر فریمورک‌ها:** شما می‌توانید DbContext را با سایر تکنولوژی‌ها، ازجمله ASP.NET Core، ترکیب کنید و ساخت وب‌اپلیکیشن‌ها را به‌همراه دسترسی به پایگاه داده انجام دهید.

چگونه از DbContext در EF Core استفاده کنیم؟

این مطلب از پاسخ به این سؤال که DbContext چیست ، فراتر می‌رود. در این بخش، چگونگی استفاده از DbContext در EF Core به‌صورت گام‌به‌گام شرح داده خواهد شد.

۱- راه اندازی DbContext

برای شروع استفاده از DbContext در اپلیکیشن EF Core ، لازم است موارد زیر را انجام دهید:

- **تعریف موجودیت‌ها:** کلاس‌هایی را ایجاد کنید که نشان‌دهنده جداول پایگاه داده شما باشند.
- **پیکربندی DbContext:** یک کلاس مشتق‌شده از DbContext ایجاد کرده و Connection String به پایگاه داده را تعیین کنید.
- **تعیین مجموعه‌های موجودیت (Entity Sets):** برای تعریف مجموعه‌های موجودیت، از خاصیت DbSet<TEntity> در کلاس DbContext خود استفاده کنید تا هر موجودیت به جدول پایگاه داده مربوط به خود نگاشت شود.

به منظور درک بهتر، به قطعه کد زیر توجه کنید:

```
public class MyDbContext : DbContext
{
    public DbSet<Product> Products { get; set; }

    protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)
    {
        optionsBuilder.UseSqlServer("connection string");
    }
}
```

برای مشاهده خروجی حاصل، کافیست کلاس MyDbContext را نمونه‌سازی (Instantiate) کنید و متدهای آن را به‌منظور ارتباط با پایگاه داده، مورد استفاده قرار دهید.

۲ -اجرای عملیات CRUD

در این مرحله، لازم است عملیات CRUD را به‌صورت زیر اجرا کنید:

- **درج موجودیت‌های جدید:** یک Enitity Object ایجاد کنید و آن را ازطریق Add یا AddRange به مجموعه موجودیت مرتبط درج کنید. درنهایت با فراخوانی SaveChanges به حفظ داده‌ها بپردازید.
- **استخراج داده‌ها:** شما می‌توانید با کمک کوئری‌های LINQ، داده‌های خاصی را از پایگاه داده فیلترسازی و استخراج کنید. ToList ، FirstOrDefault و SingleOrDefault برخی از رایج‌ترین متدها به‌شمار می‌روند.
- **به‌روزرسانی داده‌ها:** امکان ویرایش Property های مربوط به موجودیت‌های درون نمونه DbContext وجود دارد. این تغییرات به‌صورت خودکار قابل پیگیری هستند و می‌توان با فراخوانی SaveChanges، آن‌ها را Commit کرد.
- **حذف داده‌ها:** داده‌هایی را که می‌خواهید از مجموعه موجودیت یا همان Entity Set حذف شوند، استخراج کرده و آن‌ها را با کمک Remove پاک کنید. با فراخوانی SaveChanges، آن رکورد خاص به‌طور دائمی از پایگاه داده حذف خواهد شد.

برای درک موارد فوق، به قطعه کد زیر توجه کنید. این قطعه کد دارای کامنت‌های شفاف است و شما می‌توانید به‌راحتی آن را متوجه شوید.

```
// Add a new product
Product newProduct = new Product { Name = "New Product", Price = 10.99 };
dbContext.Products.Add(newProduct);
dbContext.SaveChanges();

// Retrieve all products with a price greater than $5
List<Product> expensiveProducts = dbContext.Products.Where(p => p.Price >
5).ToList();

// Update an existing product
Product productToUpdate = dbContext.Products.Find(2);
productToUpdate.Price = 12.50;
dbContext.SaveChanges();

// Delete a product
Product productToDelete = dbContext.Products.Find(3);
dbContext.Products.Remove(productToDelete);
dbContext.SaveChanges();
```

۳- استفاده از LINQ برای کوئری نویسی کارآمد

Language Integrated Query یا همان LINQ، یک کامپوننت از فریمورک .NET. است که به‌واسطه آن، می‌توانید قابلیت‌های کوئری‌نویسی برای داده‌ها را به‌صورت Native در زبان‌های برنامه‌نویسی .NET. استفاده کنید. به بیان دیگر، با کمک LINQ امکان نوشتن کوئری‌ها به‌طور مستقیم در کد سی شارپ وجود دارد. توجه به موارد زیر در این گام ضروری هستند:

- **ایجاد کوئری‌های LINQ:** براساس نیازمندی‌ها، خود را با Syntax های لازم برای فیلترسازی، مرتب‌سازی و نمایش داده‌ها آشنا کنید.
- **متدهای متداول LINQ:** مواردی همچون where، OrderBy، Skip، Takeو Select هر یک کاربردهایی برای دستکاری داده‌ها دارا هستند.

برای آشنایی بیشتر، به مثال زیر توجه کنید:

```
// Retrieve the first 10 products sorted by name
List<Product> firstTenProducts = dbContext.Products
    .OrderBy(p => p.Name)
    .Take(10)
    .ToList();

// Select only product names and prices
List<ProductDetails> productDetails = dbContext.Products
    .Select(p => new ProductDetails { Name = p.Name, Price = p.Price })
    .ToList();
```

تا این بخش از مقاله DbContext چیست ، یک نمای کلی از این مفهوم آموزش داده شد. در طول زمان و با کسب تجربه بیشتر، می‌توانید ویژگی‌های پیشرفته مختلف، همچون Lazy Loading و Eager Loading ، پیکربندی‌های سفارشی و رسیدگی به مدل‌های داده پیچیده را بررسی کنید.

ملاحظات جانبی

موارد زیر را به‌عنوان اطلاعات جانبی در نظر داشته باشید و به آن‌ها توجه کنید:

- **فرآیند Dispose نمونه‌های DbContext:** در EF Core، نمونه‌های DbContext وظیفه مدیریت Connection های پایگاه داده را برعهده دارند. در صورتی که فرآیند Dispose آن‌ها به‌درستی انجام نشود، ممکن است نشت منبع (Resource Leak) رخ دهد. در چنین شرایطی، Connection حتی زمان‌هایی که به آن نیاز نیست، Open باقی می‌ماند و احتمالاً کارایی را تحت تأثیر قرار دهد. استفاده از عبارت using برای مدیریت منابع و اجتناب از نشت احتمالی حافظه مناسب است.
- **رسیدگی به خطاها و Exception ها:** پیاده‌سازی مکانیزم‌های مناسب برای رسیدگی به خطا، باعث می‌شود که خطاهای بالقوه در طول عملیات دسترسی به داده‌ها بررسی شوند.
- **ارجاع به مستندات رسمی:** دوره‌های آموزشی می‌توانند مباحث مهم را برای شما پوشش دهند؛ با این وجود، پیشنهاد می‌شود که برای آشنایی با یک کارکرد خاص یا اطلاعات بیشتر، به مستندات مربوطه رجوع کنید.

ارتباط میان DbContext و ORM چیست؟

پیش‌تر در مقاله ORM چیست ؟ چرا از آن استفاده می کنیم؟ به بررسی نگاشت شی رابطه‌ای پرداخته‌ایم. ممکن است این سؤال برایتان پیش آمده باشد که رابطه ORM و DbContext چیست ؟ در ادامه به این پرسش پاسخ خواهیم داد. به‌طور کلی، ORM مانند رابط میان زبان برنامه‌نویسی شی‌گرا (OOP) و پایگاه داده رابطه ای (RDBMS) عمل می‌کند؛ در حالی که DbContext پیاده‌سازی الگوی ORM درون EF Core است و کارکردهای مهمی را کپسوله‌سازی می‌کند. درحقیقت، DbContext با ارائه یک رابط مناسب، پیچیدگی‌های ORM را تسهیل می‌دهد و یک رویکرد کاربردپسند برای ارتباط با دیتابیس فراهم خواهد کرد.

بهترین روش ها برای استفاده از DbContext

DbContext به توسعه‌دهندگان امکانات قدرتمند و کارآمدی در زمینه مدیریت داده‌ها ارائه می‌دهد. بهترین شیوه‌های استفاده از DbContext چیست ؟ به نکات زیر توجه کنید تا بتوانید آن را به شکل بهینه مورد استفاده قرار دهید:

- استفاده از عبارت using
- بکارگیری Dependency Injection
- استفاده از روش بارگذاری مناسب (Lazy Loading و Eager Loading)
- پیاده‌سازی الگوی Unit Of Work
- وجود مکانیزم‌های مناسب برای رسیدگی به خطا
- نگهداری و شفافیت کد
- استفاده از ابزارهای Profiling برای بررسی و بهبود کارایی

کلام پایانی: علت اهمیت DbContext چیست ؟

DbContext به‌دلیل تسهیل دسترسی به داده‌ها، بهبود نگهداری کد، کوئری‌نویسی کارآمد و توسعه سریع، نقش پراهمیتی در اپلیکیشن‌های EF Core دارا است. در این مقاله، مزایا و معایب آن را به‌همراه کاربردهای آن مورد بررسی قرار دادیم. در ادامه مطلب، به نحوه استفاده DbContext و بهترین شیوه‌های بهره‌مندی از آن اشاره کرده‌ایم. شما می‌توانید با به‌کارگیری این کلاس، از یک رابط کاربرپسند برای دسترسی به داده‌ها استفاده کنید و با پیچیدگی‌های مربوط به تعامل با دیتابیس مواجه نشوید.