بسم الله الرحمن الرحیم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

ساختن DTO ها(data transfer object)

مدرس : مهندس افشین رفوآ

دوره آموزشی Web API

در حال حاضر Web API ما نمایشی از entityها در دیتابیس به کاربر نشان می دهد. کاربران داده هایی را که به طور مستقیم به جداول دیتابیس شما وصل هستند دریافت می کنند. با این حال، این مورد همیشه یک ایده خوب به نظر نمی رسد. اغلب اوقات شما می خواهید نوع داده ای را که برای کاربر می فرستید تغییر دهید. برای مثال، شما شاید بخواهید:

- مراجع دایره ای (circular references) را حذف کنید(مقاله قبل را مطالعه کنید)
- خاصیت های خاصی که کاربر برای نمایش پشتیبانی نمی کند مخفی سازید.
- برای کمتر کردن اندازه ظرفیت انتقال چند خاصیت را حذف کنید.
- برای دسترسی راحتتر کاربران گراف های تو در تو را قابل فهم کنید.
- از معایب و مضرات ارسال های بیش از حد جلوگیری کنید.
- لایه سرویس را از لایه دیتابیس خود جدا کنید.

برای انجام دادن مطالبی که در بالا گفته شد شما می توانید یک DTO بسازید. DTO یک object است که چگونگی ارسال داده در شبکه را تعریف می کند. اجازه دهید این را با مثالی از یک entity مربوط به Book توضیح دهیم. در فولدر Models دو کلاس DTO اضافه کنید:

```
namespace BookService.Models
{
    public class BookDetailDTO
    {
        public int Id { get; set; }
        public string Title { get; set; }
        public int Year { get; set; }
        public decimal Price { get; set; }
```

```
        public string AuthorName { get; set; }
        public string Genre { get; set; }
    }
}
```

کلاس BookDetailDTO شامل همه ی خاصیت های مدل Book به جز AuthorName است که البته این مورد رشته ای برای نگه داری نام نویسنده می باشد.

سپس، دو متد GET در کلاس BookController را با ورژنی که DTO را برگرداند، جا به جا کنید. ما از دستور Select در LINQ برای تبدیل entity های Book به DTO ها استفاده کردیم.

```
// GET api/Books
public IQueryable<BookDTO> GetBooks()
{
    var books = from b in db.Books
                select new BookDTO()
                {
                    Id = b.Id,
                    Title = b.Title,
                    AuthorName = b.Author.Name
                };

    return books;
}
// GET api/Books/5
[ResponseType(typeof(BookDetailDTO))]
public async Task<IHttpActionResult> GetBook(int id)
{
    var book = await db.Books.Include(b => b.Author).Select(b =>
        new BookDetailDTO()
        {
            Id = b.Id,
            Title = b.Title,
            Year = b.Year,
            Price = b.Price,
            AuthorName = b.Author.Name,
            Genre = b.Genre
        }).SingleOrDefaultAsync(b => b.Id == id);
    if (book == null)
    {
        return NotFound();
    }
    return Ok(book);
}
```

در اینجا با استفاده از SQL یک متد جدید GetBooks می سازیم. همانطور که مشاهده می کنید ، EF دستور Select در LINQ را به یک SELECT در SQL تفسیر کرده است.

```
SELECT
    [Extent1].[Id] AS [Id],
    [Extent1].[Title] AS [Title],
```

```
    [Extent2].[Name] AS [Name]
    FROM  [dbo].[Books] AS [Extent1]
    INNER JOIN [dbo].[Authors] AS [Extent2] ON [Extent1].[AuthorId] = [Extent2].[Id]
```

در آخر نیز متد PostBook را برای برگرداندن یک DTO اصلاح می کنیم:

```
[ResponseType(typeof(Book))]
public async Task<IHttpActionResult> PostBook(Book book)
{
    if (!ModelState.IsValid)
    {
        return BadRequest(ModelState);
    }
    db.Books.Add(book);
    await db.SaveChangesAsync();
    // New code:
    // Load author name
    db.Entry(book).Reference(x => x.Author).Load();
    var dto = new BookDTO()
    {
        Id = book.Id,
        Title = book.Title,
        AuthorName = book.Author.Name
    };
    return CreatedAtRoute("DefaultApi", new { id = book.Id }, dto);
}
```