

第56期-使用Fluent方式验证数据

2019年9月13日 23:31

Fluent API 模型配置信息验证

Code First 常使用 Fluent API 配置模型，针对属性设置 Required 和 MaxLength 无法用于验证，也就是说 EF Core 没有针对 Fluent API 模式的验证，只能通过数据注解特性模式来验证。

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.Entity<Blog>().Property(p => p.Name).IsRequired().HasMaxLength(10);

    base.OnModelCreating(modelBuilder);
}
```

虽然这样，但如果你要扩展也是可以的，但不推荐，为了复习下之前的课程，我们演示下如何扩展。

```
public override int SaveChanges()
{
    var entityEntrys = ChangeTracker.Entries().Where(e => e.State == EntityState.Added || e.State == EntityState.Modified);

    foreach (EntityEntry entityEntry in entityEntrys)
    {
        IEntityType entityType = Model.FindEntityType(entityEntry.Entity.GetType().FullName);

        var validationContext = new ValidationContext(entityEntry.Entity);

        Validator.ValidateObject(entityEntry.Entity, validationContext, true);

        foreach (IProperty property in entityType.GetProperties())
        {
            var validationAttributes = new List<ValidationAttribute>();

            int? maxLength = property.GetMaxLength();

            if (maxLength.HasValue)
            {
                validationAttributes.Add(new MaxLengthAttribute(maxLength.Value));
            }

            if (!property.IsNullable)
            {
                validationAttributes.Add(new RequiredAttribute());
            }

            foreach (IAnnotation annotation in property.GetAnnotations())
            {
                var validationAttributeTypes = typeof(ValidationAttribute)
                    .Assembly.GetTypes()
                    .Where(t => t.IsSubclassOf(typeof(ValidationAttribute)));

                string attributeTypeName = string.Concat(annotation.Name, "Attribute");
```

```

        var validationAttribute = validationAttributeTypes.SingleOrDefault(t =>
t.Name.Equals(attributeTypeName));

        if (validationAttribute != null)
        {
            var attributeInstance = Activator.CreateInstance(validationAttribute,
annotation.Value) as ValidationAttribute;
            validationAttributes.Add(attributeInstance);
        }

        var currentValue = entityEntry.Property(property.Name).CurrentValue;

        Validator.ValidateValue(currentValue, validationContext, validationAttributes);
    }

    return base.SaveChanges();
}

```

使用第三方 FluentValidation 验证框架

一个轻量级第三方验证库，它使用 Fluent 形式的 Lambda 表达式来构建验证规则，该验证库可以单独使用，也可与 ASP.NET Core 深度集成实现客户端和服务端验证，关于 FluentValidation 框架的文档很多，可自行搜索引擎。

<https://github.com/JeremySkinner/FluentValidation>

```

public override int SaveChanges()
{
    var entityEntrys = ChangeTracker.Entries().Where(e => e.State == EntityState.Added || e.State ==
EntityState.Modified);

    foreach (EntityEntry entityEntry in entityEntrys)
    {
        var abstractValidatorType =
typeof(AbstractValidator<>).MakeGenericType(entityEntry.Entity.GetType());
        var modelValidatorType = Assembly.GetExecutingAssembly().GetTypes().FirstOrDefault(t =>
t.IsSubclassOf(abstractValidatorType));
        var modelValidatorInstance = (IValidator)Activator.CreateInstance(modelValidatorType);

        ValidationResult validationResult = modelValidatorInstance.Validate(entityEntry.Entity);

        if (!validationResult.IsValid)
        {
            throw new ValidationException(validationResult.Errors);
        }
    }

    return base.SaveChanges();
}

public class BlogValidator : AbstractValidator<Blog>
{
    public BlogValidator()
    {
        RuleFor(b => b.Name).Length(3, 10).Must(s => s.Contains("NB")).WithMessage("必须包含NB字符");
    }
}

```

```
    }  
}  
  
public void RunSample()  
{  
    try  
    {  
        _context.Add(new Blog { Name = "zerodo" });  
        _context.SaveChanges();  
    }  
    catch (ValidationException e)  
    {  
        e.Errors.ToList().ForEach(er => Console.WriteLine(er.ErrorMessage));  
    }  
}
```