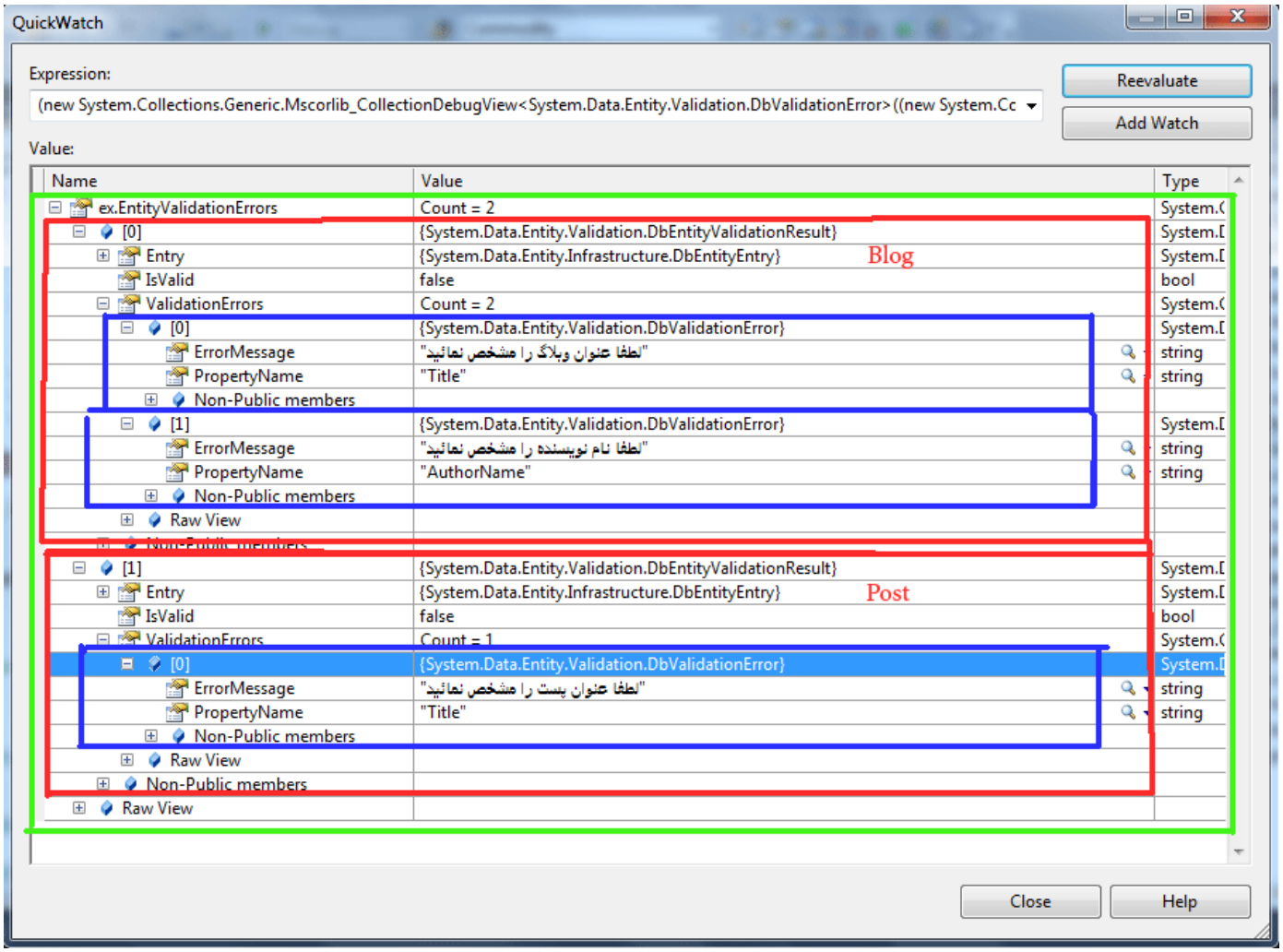
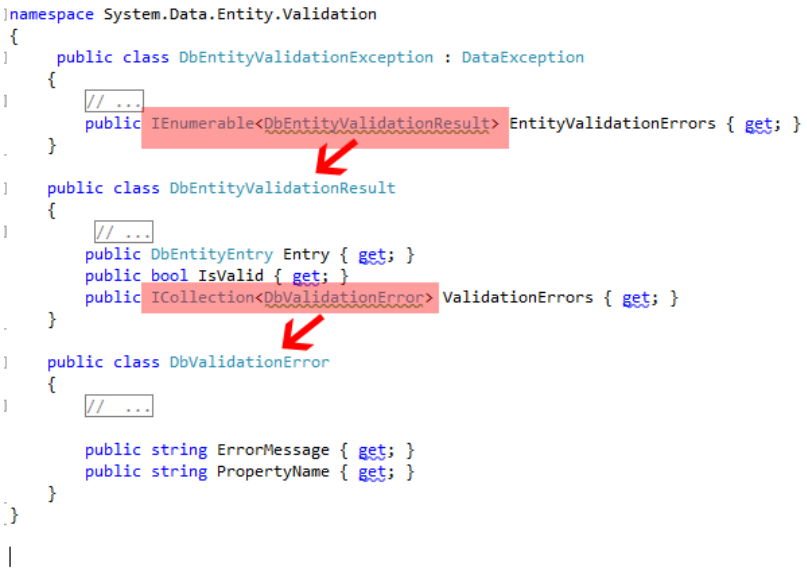
عنوان: **اعتبارسنجی در Entity framework Code first قسمت اول**
نویسنده: ایمان محمدی
تاریخ: ۱۳۹۱/۰۵/۱۳ ۱:۳
آدرس: www.dotnettips.info
گروه‌ها: Entity framework, Validation

در سری مباحث آموزشی EntityFramework وحیدی نصیری عزیز بصورت مختصر با اعتبار سنجی داده‌ها آشنا شدیم، در این آموزش سه قسمتی سعی می‌کنیم شناخت بیشتری از اعتبار سنجی داده در EF بدست بیاریم.
در EF CodeFirst بصورت پیش فرض پس از فراخوانی متد SaveChanges() اعتبار سنجی داده‌ها انجام می‌پذیرد؛ در صورتیکه که اعتبار سنجی با موفقیت انجام نشود با استثنای DbEntityValidationException روبرو می‌شویم که در اینجا از خاصیت EntityValidationErrors جهت اعلام خطاهای اعتبارسنجی استفاده می‌شود. EntityValidationErrors مجموعه‌ای از خطاهای مربوط به هر موجودیت می‌باشد و هر EntityValidationErrors شامل یک خاصیت ValidationErrors که خود مجموعه‌ای از خطاهای مربوط به propertyها می‌باشد. در تصویر زیر به خوبی می‌تونید این قضیه رو مشاهده کنید.



برای درک بهتر موضوع به ساختار کلاس‌های اعتبارسنجی در تصویر بعدی دقت کنید.



اعتبار سنجی در چه قسمت هایی اتفاق می‌افتد:

```
1.Property

2.Entity

3.Context
```

الف - Property :

در بخش سوم آموزش EF Code First با متادیتای مورد استفاده در EF و طرز استفاده از آنها آشنا شدیم.

```
[Required(ErrorMessage = "لطفا نام نویسنده را مشخص نمائید")]
        public string AuthorName { set; get; }

        [StringLength(100,MinimumLength=3,ErrorMessage="حداقل سه حرف و حداکثر 100 حرف وارد نمایید.")]
        public string AuthorName { set; get; }
```

همانطور که در ادامه می‌بینید برای اعتبار سنجی فقط به متادیتاهای واقع در DataAnnotations.dll نیاز داریم. بقیه متادیتاها مربوط به نگاشت روابط موجودیتها و نحوه ذخیره اون در دیتابیس می‌باشد.

```
Validation Attributes
AssemblySystem.ComponentModel.DataAnnotations.dll
NamespaceSystem.ComponentModel.DataAnnotations
StringLength
RegularExpression
DataType
Required
Range
CustomValidation
```

```
Mapping Attributes
AssemblyEntityFramework.dll
NamespaceSystem.ComponentModel.DataAnnotations
Key
Column,Table
ComplexType
ConCurrency
TimeStamp
DatabaseGenerated
ForeignKey
InverseProperty
MaxLength
MinLength
NotMapped
```

ب- Entity :

برای اعتبار سنجی یک موجودیت باید اینترفیس IValidatableObject پیاده سازی شود:

```
public class Blog : IValidatableObject
    {

        public int blogID { set; get; }

        [Required(ErrorMessage = "لطفا عنوان وبلاگ را مشخص نمائید")]
        public string Title { set; get; }
        [Required(ErrorMessage = "لطفا نام نویسنده را مشخص نمائید")]
        [StringLength(100,MinimumLength=3,ErrorMessage="حداقل سه حرف و حداکثر 100 حرف وارد نمایید.")]
        public string AuthorName { set; get; }


        public IEnumerable<ValidationResult> Validate(ValidationContext ValidationContext)
        {
             if (this.AuthorName =="بدون نام")
                yield return new ValidationResult
                    ("این نام برای نام نویسنده مجاز نمی‌باشد.", new[] { "AuthorName"});

            if (this.AuthorName == this.Title)
                yield return new ValidationResult
                    ("نام نویسنده وبلاگ و عنوان وبلاگ نمی‌تواند همسان باشد.!", new[] { "AuthorName",
"Title" });
        }
    }
```

نکته: در بررسی هم نام بودن نام نویسنده و نام وبلاگ هردو خاصیت ("AuthorName", "Title") رو درج کردیم اینکار باعث ایجاد دو خطای اعتبارسنجی می‌شود.

پ- Context :

برای اعتبار سنجی در سطح Context باید متد ()ValidateEntity واقع در کلاس DbContext را تحریف کنیم. این قسمت در بخش دوم مقاله کامل شرح داده خواهد شد.

```
protected override DbEntityValidationResult ValidateEntity(DbEntityEntry entityEntry,
System.Collections.Generic.IDictionary<object, object> items)
        {
            return base.ValidateEntity(entityEntry, items);
        }
```

نحوه فراخوانی اعتبار سنجی ها:

```
// اعتبار سنجی یک خاصیت
ICollection<DbValidationError> ValidationProperty = Context.Entry(Blog).Property(p =>
p.AuthorName).GetValidationErrors();

// اعتبار سنجی یک موجودیت
DbEntityValidationResult ValidationEntity = Context.Entry(Blog).GetValidationResult();

// اعتبار سنجی همه موجودیت‌ها
 IEnumerable<DbEntityValidationResult> ValidationContext = Context.GetValidationErrors();
```

نکته : در اعتبار سنجی Context بصورت پیش فرض فقط موجودیت‌های جدید و یا تغییر یافته اعتبار سنجی می‌شوند.

```
EntityState.Added ||  EntityState.Modified
```

ترتیب فراخوانی اعتبار سنجی‌ها :

ابتدا اعتبار سنجی روی Property انجام می‌گیرد در صورتی که خطایی وجود نداشته باشد اعتبار سنجی مرحله بعد یعنی موجودیت‌ها بررسی می‌شود. اگر در مرحله اعتبار سنجی خاصیت‌ها خطایی وجود داشته باشد اعتبار سنجی موجودیت انجام نمی‌گیرد. ترتیب اعتبار سنجی در مرحله Context بستگی به نحوه پیاده سازی ما دارد که در بخش دوم آموزش شرح داده خواهد شد.

سوال:
اعتبار سنجی چند زبانی رو چگونه تعریف کنیم؟

متد GetValidationErrors() رو در الگوی UOW , Repository چگونه پیاده سازی کنیم؟

آیا اعتبار سنجی در کنار موجودیت‌ها از نظر معماری چند لایه کار درستی می‌باشد؟

با پاسخ دادن به سوالات بالا در قالب نظر و یا مقاله به تکمیل موضوع کمک کنید و فضای آموزشی سایت رو رونق ببخشید.

نظرات خوانندگان

نویسنده: daneshjoo
تاریخ: ۱۳۹۲/۰۲/۰۹ ۲۱:۵۱

سلام؛ من دارم با wpf کار می‌کنم و همین طور که می‌دونید در این تکنولوژی اعتبار سنجی خوبی داره حالا سوال من اینه که چطور می‌تونم اعتبار سنجی EF 5 رو با اعتبار سنجی WPF تلفیق کنم و چیز جامع و یکپارچه ای ازش در بیاد

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۲/۰۹ ۲۲:۱۰

مراجعه کنید به قسمت پنجم سری MVVM که در مورد نحوه استفاده از data annotations برای اعتبار سنجی در WPF مطلب داره (محل دریافت دوم کل سری).

نویسنده: ایمان محمدی
تاریخ: ۱۳۹۲/۰۲/۱۰ ۸:۴۷

من از ValidationHelper که شما قرار دادید در کلاس زیر استفاده کردم و baseentity از کلاس زیر مشتق شده تا تمام موجودیت‌ها اینترفیس IDataErrorInfo رو برای wpf پیاده کرده باشند.

```
public abstract class DataErrorInfo :ObservableObject, IDataErrorInfo
    {
        [Browsable(false)]
        public string Error
        {
            get
            {
                var errors = ValidationHelper.GetErrors(this);
                return string.Join(Environment.NewLine, errors);
            }
        }

        public string this[string columnName]
        {
            get
            {
                var errors = ValidationHelper.ValidateProperty(this, columnName);
                return string.Join(Environment.NewLine, errors);
            }
        }
    }
```

نویسنده: ایمان محمدی
تاریخ: ۱۳۹۲/۰۲/۲۲ ۰:۵۲

ValidationHelper مورد نیاز جهت کلاس DataErrorInfo :

```
public class ValidationHelper
    {
        public static IList<ValidationResult> GetErrors(object instance)
        {
            var res = new List<ValidationResult>();
            Validator.TryValidateObject(instance,
                new ValidationContext(instance, null, null), res, true);
            return res;
        }



        public static IList<ValidationResult> ValidateProperty(object value, string propertyName)
        {
```

```
            if (string.IsNullOrEmpty(propertyName))
                throw new ArgumentException("Invalid property name", propertyName);

            var propertyValue = value.GetType().GetProperty(propertyName).GetValue(value, null);

            var results = new List<ValidationResult>();
            var context = new ValidationContext(value, null, null) { MemberName = propertyName };
            Validator.TryValidateProperty(propertyValue, context, results);
            return results;
        }
    }
```

نویسنده: ایمان
تاریخ: ۱۳۹۲/۱۱/۱۴ ۱۲:۲۹

سلام خسته نباشید

قسمت دوم این مقاله رو ننوشتین تو سایت؟

خیلی خوبه و من واقعا بهش احتیاج دارم چون دارم الگوی Repositor,Uow رو پیاده سازی میکنم و تو اعتبار سنجی هاش به مشکل خوردم و نمیدونم تو یه بیزینس بزرگ چجوری باید اون رو پیاده سازی کرد که به مشکل نخوره

نویسنده: ابوالفضل موسوی
تاریخ: ۱۳۹۳/۰۴/۱۹ ۱۶:۱۸

سلام . یه سوال داشتم . توی codefirst می‌خوام فیلدهای جدول تولید شده نالیبل باشند و از اعتبار سنجی سمت کلاینت و سرور(MVC) هم استفاده کنم [Requierd] . اگه امکانش هست راهنمای کنید ؟

نویسنده: محسن خان
تاریخ: ۱۳۹۳/۰۴/۱۹ ۱۶:۳۵

از ViewModel باید استفاده و اعتبارسنجی رو به ViewModel اعمال کنید.

نویسنده: ابوالفضل موسوی
تاریخ: ۱۳۹۳/۰۴/۲۰ ۱۲:۴۹

این کاری که شما گفتین دوباره کاریه! و فیلدهای که من دارم و همچنین جداول خیلی زیادن. من تونستم NULL بودن و اعتبار سنجی سمت سرور رو انجام بدم؛ با کدهایی که زیر قرار میدم . اما چطوری با جی کوری ایجکس باید این ولیدیشنو سمن کلاینت نیز فرا خوامی بکنم ؟

```
[AttributeUsage(AttributeTargets.Field | AttributeTargets.Property, AllowMultiple = false, Inherited =
true)]
    public class RequierdExAttribute : ValidationAttribute
    {
        public RequierdExAttribute(string ErrorMessage)
            : base()
        {
            this.ErrorMessage = ErrorMessage;
        }

        public override bool IsValid(object value)
        {
            if (value == null) return false;
```

```
            return true;
        }
```

حالا بجایِ Requierd روی فیلدها از RequierdِEx استفاده میکنم که فیلد مورد نظر در دیتا بیس نال پذیر ساخته میشه . اما مشکل اعتبار سنجی سمت کلاینته؟

نویسنده: محسن خان
تاریخ: ۱۳۹۳/۰۴/۲۰ ۱۳:۰

- کاری که می‌خوای، منطقا زیر سؤاله. هم قرار نال پذیر باشه. هم کاربر باید اجبارا پرش کنه! یعنی چی اینکار؟!

- در مورد ویژگی‌های اعتبار سنجی سفارشی و مدیریت کدهای سمت کلاینت اون‌ها مطلب در سایت موجوده:

طراحی ValidationAttribute دلخواه و هماهنگ سازی آن با ASP.NET MVC

MVC 13 قسمت اعتبارسنجی سفارشی

نویسنده: ابوالفضل موسوی
تاریخ: ۱۳۹۳/۰۴/۲۰ ۱۵:۴۰

تا حالا شده که شما بخوای یه فیلد از جدولو در یه مقطعی پر کنی بعد فیلدهای بعدی رو در جاهای دیگه پر کنید ؟ خوب وقتی مقدار فیلدها نال پذیر نباشه نمیشه در مرحله ای اول اون فیلدو ذخیره کرد ! حالا چون ولیدیشن‌های MVC رو هم نیاز دارم پس باید requird هم باشه فیلد ها... تا در مراحل بعدی بگم کدام فیلدها ورودشون اجباریه ! منطقش درسته ! این کاری که می‌گم برای ذخیره سازی چند جدول به صورت همزمانه ! که چند جدول با هم ارتباط هم دارن ! توی database first کار میکنه اما توی code first به دلیل این مشکل کار نمی‌کرد ! من فقط می‌خوام وقتی داده‌ی در جدول اولی ذخیره شد ایدی اون در جدول دومی ذخیره بشه به عنوان کلید خارجی بعد اطلاعات دیگه بعدا پر بشه ! همین ! مرسی از لینک‌های که قرارا دادین برسی کردم لینک دوم کاری که من انجام دادمو اورده سمت کلاینت . و تقریبا مشکلم حل شد . تشکر