عنوان: **انتقال خودکار Data Annotations از مدل‌ها به ViewModelهای ASP.NET MVC به کمک AutoMapper**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۲/۲۷ ۱۷:۳۴
آدرس: www.dotnettips.info
گروه‌ها: MVC, Entity framework, AutoMapper

عموما مدل‌های ASP.NET MVC یک چنین شکلی را دارند:

```
public class UserModel
{
    public int Id { get; set; }

    [Required(ErrorMessage = "(*)")]
    [Display(Name = "نام")]
    [StringLength(maximumLength: 10, MinimumLength = 3, ErrorMessage = "نام باید حداقل 3 و حداکثر 10 حرف باشد")]
    public string FirstName { get; set; }

    [Required(ErrorMessage = "(*)")]
    [Display(Name = "نام خانوادگی")]
    [StringLength(maximumLength: 10, MinimumLength = 3, ErrorMessage = "نام خانوادگی باید حداقل 3 و حداکثر 10 حرف باشد")]
    public string LastName { get; set; }
}
```

و ViewModel مورد استفاده برای نمونه چنین ساختاری را دارد:

```
public class UserViewModel
{
      public string FirstName { get; set; }
      public string LastName { get; set; }
}
```

مشکلی که در اینجا وجود دارد، نیاز به کپی و تکرار تک تک ویژگی‌های (Data Annotations/Attributes) خاصیت‌های مدل، به خواص مشابه آن‌ها در ViewModel است؛ از این جهت که می‌خواهیم برچسب خواص ViewModel، از ویژگی Display دریافت شوند و همچنین اعتبارسنجی‌های فیلدهای اجباری و بررسی حداقل و حداکثر طول فیلدها نیز حتما اعمال شوند (هم در سمت کاربر و هم در سمت سرور).
در ادامه قصد داریم راه حلی را به کمک جایگزین سازی Providerهای توکار ASP.NET MVC با نمونه‌ی سازگار با AutoMapper، ارائه دهیم، به نحوی که دیگر نیازی نباشد تا این ویژگی‌ها را در ViewModelها تکرار کرد.

قسمت‌هایی از ASP.NET MVC که باید جهت انتقال خودکار ویژگی‌ها تعویض شوند

ASP.NET MVC به صورت توکار دارای یک ModelMetadataProviders.Current است که از آن جهت دریافت ویژگی‌های هر خاصیت استفاده می‌کند. می‌توان این تامین کننده‌ی ویژگی‌ها را به نحو ذیل سفارشی سازی نمود.
در اینجا IConfigurationProvider همان Mapper.Engine.ConfigurationProvider مربوط به AutoMapper است. از آن جهت استخراج اطلاعات نگاشت‌های AutoMapper استفاده می‌کنیم. برای مثال کدام خاصیت Model به کدام خاصیت ViewModel نگاشت شده‌است. این‌کارها توسط متد الحاقی GetMappedAttributes انجام می‌شوند که در ادامه‌ی مطلب معرفی خواهد شد.

```
public class MappedMetadataProvider : DataAnnotationsModelMetadataProvider
{
    private readonly IConfigurationProvider _mapper;

    public MappedMetadataProvider(IConfigurationProvider mapper)
    {
        _mapper = mapper;
    }

    protected override ModelMetadata CreateMetadata(
        IEnumerable<Attribute> attributes,
        Type containerType,
        Func<object> modelAccessor,
        Type modelType,
        string propertyName)
```

```
    {
        var mappedAttributes =
            containerType == null ?
            attributes :
            _mapper.GetMappedAttributes(containerType, propertyName, attributes.ToList());
        return base.CreateMetadata(mappedAttributes, containerType, modelAccessor, modelType,
propertyName);
    }
}
```

شبیه به همین کار را باید برای ModelValidatorProviders.Providers نیز انجام داد. در اینجا یکی از تامین کننده‌های ModelValidator، از نوع DataAnnotationsModelValidatorProvider است که حتما نیاز است این مورد را نیز به نحو ذیل سفارشی سازی نمود. در غیراینصورت error messages موجود در ویژگی‌های تعریف شده، به صورت خودکار منتقل نخواهند شد.

```
public class MappedValidatorProvider : DataAnnotationsModelValidatorProvider
{
    private readonly IConfigurationProvider _mapper;

    public MappedValidatorProvider(IConfigurationProvider mapper)
    {
        _mapper = mapper;
    }

    protected override IEnumerable<ModelValidator> GetValidators(
        ModelMetadata metadata,
        ControllerContext context,
        IEnumerable<Attribute> attributes)
    {

        var mappedAttributes =
            metadata.ContainerType == null ?
            attributes :
            _mapper.GetMappedAttributes(metadata.ContainerType, metadata.PropertyName,
attributes.ToList());
        return base.GetValidators(metadata, context, mappedAttributes);
    }
}
```

و در اینجا پیاده سازی متد GetMappedAttributes را ملاحظه می‌کنید.

ASP.NET MVC هر زمانیکه قرار است توسط متدهای توکار خود مانند Html.TextBoxFor, Html.ValidationMessageFor، اطلاعات خاصیت‌ها را تبدیل به المان‌های HTML کند، از تامین کننده‌های فوق جهت دریافت اطلاعات ویژگی‌های مرتبط با هر خاصیت استفاده می‌کند. در اینجا فرصت داریم تا ویژگی‌های مدل را از تنظیمات AutoMapper دریافت کرده و سپس بجای ویژگی‌های خاصیت معادل ViewModel درخواست شده، بازگشت دهیم. به این ترتیب ASP.NET MVC تصور خواهد کرد که ViewModel ما نیز دقیقا دارای همان ویژگی‌های Model است.

```
public static class AutoMapperExtensions
{
    public static IEnumerable<Attribute> GetMappedAttributes(
        this IConfigurationProvider mapper,
        Type viewModelType,
        string viewModelPropertyName,
        IList<Attribute> existingAttributes)
    {
        if (viewModelType != null)
        {
            foreach (var typeMap in mapper.GetAllTypeMaps().Where(i => i.DestinationType ==
viewModelType))
            {
                var propertyMaps = typeMap.GetPropertyMaps()
                    .Where(propertyMap => !propertyMap.IsIgnored() && propertyMap.SourceMember != null)
                    .Where(propertyMap => propertyMap.DestinationProperty.Name ==
viewModelPropertyName);

                foreach (var propertyMap in propertyMaps)
                {
                    foreach (Attribute attribute in propertyMap.SourceMember.GetCustomAttributes(true))
                    {
                        if (existingAttributes.All(i => i.GetType() != attribute.GetType()))
                        {
                            yield return attribute;
```

```
                        }
                    }
                }
            }
        }

        if (existingAttributes == null)
        {
            yield break;
        }

        foreach (var attribute in existingAttributes)
        {
            yield return attribute;
        }
    }
}
```

ثبت تامین کننده‌های سفارشی سازی شده توسط AutoMapper

پس از تهیه‌ی تامین کننده‌های انتقال ویژگی‌ها، اکنون نیاز است آن‌ها را به ASP.NET MVC معرفی کنیم:

```
protected void Application_Start()
{
    AreaRegistration.RegisterAllAreas();
    WebApiConfig.Register(GlobalConfiguration.Configuration);
    FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
    RouteConfig.RegisterRoutes(RouteTable.Routes);

    Mappings.RegisterMappings();
    ModelMetadataProviders.Current = new MappedMetadataProvider(Mapper.Engine.ConfigurationProvider);

    var modelValidatorProvider = ModelValidatorProviders.Providers
        .Single(provider => provider is DataAnnotationsModelValidatorProvider);
    ModelValidatorProviders.Providers.Remove(modelValidatorProvider);
    ModelValidatorProviders.Providers.Add(new
MappedValidatorProvider(Mapper.Engine.ConfigurationProvider));
}
```

در اینجا ModelMetadataProviders.Current با MappedMetadataProvider جایگزین شده‌است.

در قسمت کار با ModelValidatorProviders.Providers، ابتدا صرفا همان تامین کننده‌ی از نوع DataAnnotationsModelValidatorProvider پیش فرض، یافت شده و حذف می‌شود. سپس تامین کننده‌ی سفارشی سازی شده‌ی خود را معرفی می‌کنیم تا جایگزین آن شود.

مثالی جهت آزمایش انتقال خودکار ویژگی‌های مدل به ViewModel

کنترلر مثال برنامه به شرح زیر است. در اینجا از متد Mapper.Map جهت تبدیل خودکار مدل کاربر به ViewModel آن استفاده شده‌است:

```
public class HomeController : Controller
{
    public ActionResult Index()
    {
        var model = new UserModel { FirstName = "و", Id = 1, LastName = "ن" };
        var viewModel = Mapper.Map<UserViewModel>(model);
        return View(viewModel);
    }

    [HttpPost]
    public ActionResult Index(UserViewModel data)
    {
        return View(data);
    }
}
```

با این View که جهت ثبت اطلاعات مورد استفاده قرار می‌گیرد. این View، اطلاعات مدل خود را از ViewModel معرفی شده‌ی در ابتدای بحث دریافت می‌کند:

```
@model Sample12.ViewModels.UserViewModel

@using (Html.BeginForm("Index", "Home", FormMethod.Post, htmlAttributes: new { @class = "form-
horizontal", role = "form" }))
{
    <div class="row">
        <div class="form-group">
            @Html.LabelFor(d => d.FirstName, htmlAttributes: new { @class = "col-md-2 control-label" })
            <div class="col-md-10">
                @Html.TextBoxFor(d => d.FirstName)
                @Html.ValidationMessageFor(d => d.FirstName)
            </div>
        </div>
        <div class="form-group">
            @Html.LabelFor(d => d.LastName, htmlAttributes: new { @class = "col-md-2 control-label" })
            <div class="col-md-10">
                @Html.TextBoxFor(d => d.LastName)
                @Html.ValidationMessageFor(d => d.LastName)
            </div>
        </div>
        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="ارسال" class="btn btn-default" />
            </div>
        </div>
    </div>
}
```

در این حالت اگر برنامه را اجرا کنیم به شکل زیر خواهیم رسید:

در این شکل هر چند نوع مدل View مورد استفاده از ViewModel ایی تامین شده‌است که دارای هیچ ویژگی Data Annotations/Attributes نیست، اما برچسب هر فیلد از ویژگی Display دریافت شده‌است. همچنین اعتبارسنجی سمت کاربر فعال بوده و برچسب‌های آن‌ها نیز به درستی دریافت شده‌اند.

کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید .

نظرات خوانندگان

نویسنده: سیدمجتبی حسینی
تاریخ: ۱۳۹۴/۰۳/۱۷ ۱۲:۳۳

سلام
با توجه به بخش Other Notes در این مطلب استفاده همزمان از انتقال خودکار Data Annotations و تزریق وابستگی‌های AutoMapper در لایه سرویس برنامه چگونه است؟

متشکرم

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۳/۱۷ ۱۲:۳۹

به چه مشکلی برخوردید؟ کار نکرد؟ خطا داد؟ چه خطایی داد؟ چطور استفاده کردید؟

نویسنده: سیدمجتبی حسینی
تاریخ: ۱۳۹۴/۰۳/۱۷ ۱۲:۴۱

بخش تزریق وابستگی به خوبی کار میکند اما بخش انتقال خودکار Data Annotations عمل نمی‌کند و انتقال صورت نمی‌گیرد.
علیرغم اینکه تمام بخش‌های آن اجرا می‌شود.
توالی کدهای مربوط در global.asax بصورت زیر است:

```
setDbInitializer();
ModelMetadataProviders.Current = new MappedMetadataProvider(Mapper.Engine.ConfigurationProvider);
var modelValidatorProvider = ModelValidatorProviders.Providers
      .Single(provider => provider is DataAnnotationsModelValidatorProvider);
ModelValidatorProviders.Providers.Remove(modelValidatorProvider);
ModelValidatorProviders.Providers.Add(new
MappedValidatorProvider(Mapper.Engine.ConfigurationProvider));
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۳/۱۷ ۱۲:۴۹

همینطور هست. علت آن‌را در نظرات مطلب تزریق وابستگی‌های AutoMapper توضیح دادم:
«کاری که در اینجا انجام شده، ایجاد یک Mapping Engine *سفارشی* هست که با Mapping Engine اصلی استاتیک یکی نیست. به همین جهت برای نمونه متد Project آرگومان (**_mappingEngine**)Project هم دارد. اگر قید نشود، یعنی قرار است از موتور نگاشت استاتیک سراسری پیش فرض آن استفاده شود. »
در مثال فوق هم Mapper.Engine.ConfigurationProvider از همان موتور نگاشت استاتیک سراسری استفاده شده‌است (در متد Application_Start برنامه). این مورد را باید با یک وهله‌ی IConfigurationProvider تامین شده‌ی توسط IoC Container، جایگزین کنید؛ مثلا:

```
SmObjectFactory.Container.GetInstance<IConfigurationProvider>()
```

نویسنده: سیدمجتبی حسینی
تاریخ: ۱۳۹۴/۰۳/۱۷ ۱۲:۵۵

متشکرم مشکل حل شد.
استفاده از این وهله در Application_Start مشکل ساز نیست؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۳/۱۷ ۱۲:۵۷

IConfigurationProvider وابستگی به ASP.NET ندارد و همچنین طول عمر ConfigurationStore آن هم Singleton است و یکبار که ایجاد شد، کش می‌شود.