عنوان: **استفاده ازExpressionها جهت ایجاد Strongly typed view در ASP.NET MVC**
نویسنده: محسن موسوی
تاریخ: ۱۳۹۳/۰۴/۱۵ ۰:۰
آدرس: www.dotnettips.info
گروه‌ها: MVC, Reflection, Kendo UI, jqGrid

مدل زیر را در نظر بگیرید:

```
/// <summary>
    ///
    /// </summary>
    public class CompanyModel
    {
        /// <summary>
        /// Table Identity
        /// </summary>
        public int Id { get; set; }

        /// <summary>
        /// Company Name
        /// </summary>
        [DisplayName("نام شرکت")]
        public string CompanyName { get; set; }

        /// <summary>
        /// Company Abbreviation
        /// </summary>
        [DisplayName("نام اختصاری شرکت")]
        public string CompanyAbbr { get; set; }

    }
```

از View زیر جهت نمایش لیستی از شرکت‌ها متناظر با مدل جاری استفاده میشود:

```
@{
    const string viewTitle = "شرکت ها";
    ViewBag.Title = viewTitle;
    const string gridName = "companies-grid";
}
<div class="col-md-12">
    <div class="form-panel">
        <header>
            <div class="title">
                <i class="fa fa-book"></i>
                @viewTitle
            </div>
        </header>
        <div class="panel-body">
            <div id="@gridName">
            </div>
        </div>
    </div>
</div>
</div>
@section scripts
{
    <script type="text/javascript">
        $(document).ready(function () {
            $("#@gridName").kendoGrid({
                dataSource: {
                    type: "json",
                    transport: {
                        read: {
                            url: "@Html.Raw(Url.Action(MVC.Company.CompanyList()))",
                            type: "POST",
                            dataType: "json",
                            contentType: "application/json"
                        }
                    },
                    schema: {
                        data: "Data",
                        total: "Total",
                        errors: "Errors"
                        }
                    },
                    pageSize: 10,
```

```
                    serverPaging: true,
                    serverFiltering: true,
                    serverSorting: true
                },
                pageable: {
                    refresh: true
                },
                sortable: {
                    mode: "multiple",
                    allowUnsort: true
                },
                editable: false,
                filterable: false,
                scrollable: false,
                columns: [ {
                    field: "CompanyName",
                    title: "نام شرکت",
                    sortable: true,
                }, {
                    field: "CompanyAbbr",
                    title: "مخفف نام شرکت",
                    sortable: true
                }]
            });
        });
    </script>
}
```

مشکلی که در کد بالا وجود دارد این است که با تغییر نام هر یک از متغییر هایمان ، اطلاعات گرید در ستون مربوطه نمایش داده نمیشود.همچنین عناوین ستونها نیز از DisplayName مدل پیروی نمیکنند.توسط متدهای الحاقی زیر این مشکل برطرف شده است.

```
/// <summary>
    ///
    /// </summary>
    public static class PropertyExtensions
    {
        /// <summary>
        ///
        /// </summary>
        /// <typeparam name="T"></typeparam>
        /// <param name="expression"></param>
        /// <returns></returns>
        public static MemberInfo GetMember<T>(this Expression<Func<T, object>> expression)
        {
            var mbody = expression.Body as MemberExpression;

            if (mbody != null) return mbody.Member;
            //This will handle Nullable<T> properties.
            var ubody = expression.Body as UnaryExpression;
            if (ubody != null)
            {
                mbody = ubody.Operand as MemberExpression;
            }
            if (mbody == null)
            {
                throw new ArgumentException("Expression is not a MemberExpression", "expression");
            }
            return mbody.Member;
        }

        /// <summary>
        ///
        /// </summary>
        /// <typeparam name="T"></typeparam>
        /// <param name="expression"></param>
        /// <returns></returns>
        public static string PropertyName<T>(this Expression<Func<T, object>> expression)
        {
            return GetMember(expression).Name;
        }

        /// <summary>
        ///
        /// </summary>
```

```
        /// <typeparam name="T"></typeparam>
        /// <param name="expression"></param>
        /// <returns></returns>
        public static string PropertyDisplay<T>(this Expression<Func<T, object>> expression)
        {
            var propertyMember = GetMember(expression);
            var displayAttributes = propertyMember.GetCustomAttributes(typeof(DisplayNameAttribute),
true);
            return displayAttributes.Length == 1 ?
((DisplayNameAttribute)displayAttributes[0]).DisplayName : propertyMember.Name;
        }
    }
```

```
public static string PropertyName<T>(this Expression<Func<T, object>> expression)
```

جهت بدست آوردن نام متغییر هایمان استفاده مینماییم.

```
public static string PropertyDisplay<T>(this Expression<Func<T, object>> expression)
```

جهت بدست آوردن DisplayNameAttribute استفده میشود. درصورتیکه این DisplayNameAttribute یافت نشود نام متغییر بازگشت داده میشود.

بنابراین View مربوطه را اینگونه بازنویسی میکنیم:

```
@using Models
@{
    const string viewTitle = "شرکت ها";
    ViewBag.Title = viewTitle;
    const string gridName = "companies-grid";
}
<div class="col-md-12">
    <div class="form-panel">
        <header>
            <div class="title">
                <i class="fa fa-book"></i>
                @viewTitle
            </div>
        </header>
        <div class="panel-body">
            <div id="@gridName">
            </div>
        </div>
    </div>
</div>
</div>
@section scripts
{
    <script type="text/javascript">
        $(document).ready(function () {
            $("#@gridName").kendoGrid({
                dataSource: {
                    type: "json",
                    transport: {
                        read: {
                            url: "@Html.Raw(Url.Action(MVC.Company.CompanyList()))",
                            type: "POST",
                            dataType: "json",
                            contentType: "application/json"
                        }
                    },
                    schema: {
                        data: "Data",
                        total: "Total",
                        errors: "Errors"
                        }
```

```
                    },
                    pageSize: 10,
                    serverPaging: true,
                    serverFiltering: true,
                    serverSorting: true
                },
                pageable: {
                    refresh: true
                },
                sortable: {
                    mode: "multiple",
                    allowUnsort: true
                },
                editable: false,
                filterable: false,
                scrollable: false,
                columns: [ {
                    field: "@(PropertyExtensions.PropertyName<CompanyModel>(a => a.CompanyName))",
                    title: "@(PropertyExtensions.PropertyDisplay<CompanyModel>(a => a.CompanyName))",
                    sortable: true,
                }, {
                    field: "@(PropertyExtensions.PropertyName<CompanyModel>(a => a.CompanyAbbr))",
                    title: "@(PropertyExtensions.PropertyDisplay<CompanyModel>(a => a.CompanyAbbr))",
                    sortable: true
                }]
            });
        });
    </script>
}
```

نظرات خوانندگان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۱۶ ۱۲:۳۸

با تشکر از شما. حالت پیشرفته‌تر این مساله، کار با مدل‌های تو در تو هست. برای مثال:

```
public class CompanyModel
    {
        public int Id { get; set; }
        public string CompanyName { get; set; }
        public string CompanyAbbr { get; set; }

        public Product Product { set; get; }
    }

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

در اینجا اگر بخواهیم Product.Id را بررسی کنیم:

```
var data = PropertyExtensions.PropertyName<CompanyModel>(x => x.Product.Id);
```

فقط Id آن دریافت می‌شود.
راه حلی که از کدهای EF برای این مساله استخراج شده به صورت زیر است (نمونه‌اش متد Include تو در تو بر روی چند خاصیت):

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Linq.Expressions;

namespace PropertyExtensionsApp
{
    public class PropertyHelper : ExpressionVisitor
    {
        private Stack<string> _stack;
        public string GetNestedPropertyPath(Expression expression)
        {
            _stack = new Stack<string>();
            Visit(expression);
            return _stack.Aggregate((s1, s2) => s1 + "." + s2);
        }

        protected override Expression VisitMember(MemberExpression expression)
        {
            if (_stack != null)
                _stack.Push(expression.Member.Name);
            return base.VisitMember(expression);
        }

        public string GetNestedPropertyName<TEntity>(Expression<Func<TEntity, object>> expression)
        {
            return GetNestedPropertyPath(expression);
        }
    }
}
```

در این حالت خواهیم داشت:

```
var name = new PropertyHelper().GetNestedPropertyName<CompanyModel>(x => x.Product.Id);
```

که خروجی Product.Id را بر می‌گرداند.

نویسنده: محسن موسوی
تاریخ: ۱۳۹۳/۰۷/۲۶ ۱۸:۸

در نهایت این متد به این شکل اصلاح شود:

```
/// <summary>
        ///
        /// </summary>
        /// <typeparam name="T"></typeparam>
        /// <param name="expression"></param>
        /// <returns></returns>
        public static string PropertyName<T>(this Expression<Func<T, object>> expression)
        {
            return new PropertyHelper().GetNestedPropertyName(expression);
        }
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۹/۱۰ ۲۳:۵۷

روش دیگری در اینجا: « Strongly-Typed ID References to Razor-Generated Fields »

نویسنده: محسن موسوی
تاریخ: ۱۳۹۳/۰۹/۱۱ ۱۴:۱

با تشکر
- نظر نویسنده مقاله تغییر کرده، بدلیل دوباره کاری انجام شده.(توضیحات بیشتر در کامنتهای مقاله ارجاعی)
- روش جاری وابسته به مدل ویو نیست و به همین دلیل محدودیت نداره، بطور مثال یک ویو شامل عملیات‌های اضافه و ویرایش و حذف و گرید لیست آنهاست.