عنوان: **به دست آوردن اطلاعات کد اجراکننده یک متد**
نویسنده: سیدمجتبی حسینی
تاریخ: ۱۳۹۱/۰۷/۱۸ ۰:۷
آدرس: www.dotnettips.info
برچسب‌ها: C#, Attributes

در C# 5 به بعد می‌توان به پارامترهای یک متد، پارامترهای دلخواهی را افزود تا به واسطه آن‌ها مشخصات کدی که این متد را فراخوانده، به دست آورد. روش انجام این کار، افزودن صفات زیر به پارامترهای متد مورد نظر است:
[CallerFilePath]:مسیر کد فراخواننده را نگه می‌دارد.

[CallerLineNumber]: شماره خط کد فراخواننده را نگه می‌دارد.

[CallerMemberName] : نام کد فراخوان را نگه می‌دارد .

این صفات کامپایلر را قادر می‌سازد که اطلاعاتی درباره فراخواننده متد مورد نظر، جمع‌آوری کند
مثال زیر را در نظر بگیرید:

```
using System;
using System.Runtime.CompilerServices;
namespace ConsoleApplication8
{
  class Program
    {
        static void Main(string[] args)
        {
            Test();
            Console.Read();
        }
        static void Test(
                [CallerMemberName] string memberName = null,
                [CallerFilePath] string filePath = null,
                [CallerLineNumber] int lineNumber = 0)
        {
            Console.WriteLine(memberName);
            Console.WriteLine(filePath);
            Console.WriteLine(lineNumber);
        }
    }
}
```

که نتیجه اجرای کد فوق به صورت زیر است:

```
Main
c:\Pojects\ConsoleApplication8\Program.cs
9
```

که عبارت Main عنوان متدی است که محل فراخوانی متد مورد نظر ماست و خط دوم حاوی مسیری است که کد فراخواننده متد مورد نظر ما درآن‌جا ذخیره شده است و عدد 9 نشانگر شماره خط محل فراخوانی متد Test است.

عنوان: **اعتبارسنجی در فرم‌های ASP.NET MVC با Remote Validation**
نویسنده: علی اکبر کورش فر
تاریخ: ۱۳۹۳/۰۵/۱۱ ۱۲:۵۵
آدرس: www.dotnettips.info
گروه‌ها: MVC, Validation, Attributes

بعد از آمدن نسخه‌ی سوم ASP.NET MVC مکانیسمی به نام Remote Validation به آن اضافه شد که کارش اعتبارسنجی از راه دور بود. فرض کنید نیاز است در یک فرم، قبل از اینکه کل فرم به سمت سرور ارسال شود، مقداری بررسی شده و اعتبارسنجی آن انجام گیرد و این اعتبارسنجی چیزی نیست که بتوان سمت کاربر و بدون فرستاده شدن مقداری به سمت سرور صورت گیرد.
نمونه بارز این مسئله صفحه عضویت اکثر سایت‌هایی هست که روزانه داریم با آن‌ها کار می‌کنیم. فیلد نام کاربری توسط شما پر شده و بعد از بیرون آمدن از آن فیلد، سریعا مشخص می‌شود که آیا این نام کاربری قابل استفاده برای شما هست یا خیر.
به‌صورت معمول برای انجام این کار باید با جاوا اسکریپت، مدیریتی روی فیلد مربوطه انجام دهیم. مثلا با بیرون آمدن فوکوس از روی فیلد، با Ajax نام کاربری وارد شده را به سمت سرور بفرستیم، چک کنیم و بعد از اینکه جواب برگشت بررسی کنیم که الان آیا این نام کاربری قبلا گرفته شده یا نه.
انجام این کار به‌راحتی با مزین‌کردن خصوصیت (Property) مربوطه موجود در مدل برنامه به Attribute یا ویژگی Remote و داشتن یک Action در Controller مربوطه که کارش بررسی وجود یوزرنیم هست امکان پذیر است. ادامه بحث را با مثال همراه می‌کنم.
به عنوان مثال در سیستمی که قرار هست محصولات ما را ثبت کند، باید بیایم و قبل از اینکه محصول جدید به ثبت برسد این عملیات چک‌کردن را انجام دهیم تا کالای تکراری وارد سیستم نشود. شناسه اصلی که برای هر محصول وجود دارد بارکد هست و ما آن را میخواهیم مورد بررسی قرار دهیم.

مدل برنامه

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

        [Display(Name = "نام کالا")]
        [Required(ErrorMessage = "{0} یک فیلد اجباری است و باید آن را وارد کنید.")]
        [StringLength(50, ErrorMessage = "طول {0} باید کمتر از {1} کاراکتر باشد.")]
        public string Name { get; set; }

        [Display(Name = "قیمت")]
        [Required(ErrorMessage = "{0} یک فیلد اجباری است و باید آن را وارد کنید.")]
        [DataType(DataType.Currency)]
        public double Price { get; set; }

        [Display(Name = "بارکد")]
        [Required(ErrorMessage = "{0} یک فیلد اجباری است و باید آن را وارد کنید.")]
        [StringLength(50, ErrorMessage = "طول {0} باید کمتر از {1} کاراکتر باشد.")]
        [Remote("IsProductExist", "Product", HttpMethod = "POST", ErrorMessage = "این بارکد از قبل در
سیستم وجود دارد.")]
        public string Barcode { get; set; }
    }
```

همونطور که می‌بینید خصوصیت Barcode را مزین کردیم به ویژگی Remote. این ویژگی دارای ورودی‌های خاص خودش هست. وارد کردن نام اکشن و کنترلر مربوطه برای انجام این چک‌کردن از مهم‌ترین قسمت‌های اصلی هست. چیزهایی دیگه‌ای هم هست که می‌توانیم آن‌ها را مقداردهی کنیم. مثل HttpMethod، ErrorMessage و یا AdditionFields. HttpMethod که همان طریقه‌ی ارسال درخواست به سرور هست. ErrorMessage هم همان خطایی هست که در زمان رخداد قرار است نشان داده شود. AdditionFields هم خصوصیتی را مشخص می‌کند که ما می‌خوایم به‌همراه فیلد مربوطه به سمت سرور بفرستیم. مثلا می‌تونیم به‌همراه بارکد، نام کالا را هم برای بررسی‌های مورد نیازمان بفرستیم.

کنترلر برنامه

```
[HttpPost]
        [OutputCache(Location = OutputCacheLocation.None, NoStore = true)]
        public ActionResult IsProductExist(string barcode)
```

```
        {
            if (barcode == "123456789") return Json(false); // اگر محصول وجود داشت
            return Json(true);
        }
```

در اینجا به نمایش قسمتی از کنترلر برنامه می‌پردازیم. اکشنی که مربوط می‌شود به چک‌کردن مقدارهای لازم و در پایان آن یک خروجی Json را برمی‌گردانیم که مقدار true یا false دارد. در حقیقت مقدار را به این صورت برمی‌گردانیم که اگر مقدار ورودی در پایگاه داده وجود دارد، false را برمی‌گرداند و اگر وجود نداشت true. همین‌طور آمدیم از کش شدن درخواست‌هایی که با Ajax آمده با ویژگی OutputCache جلوگیری کردیم.

نظرات خوانندگان

نویسنده: مجتبی
تاریخ: ۱۳۹۳/۰۵/۱۲ ۱۳:۳

مشکل این روش این است که بعد از یک بار اعتبار سنجی دفعه بعد با زدن هر کلید داخل تکست باکس می‌خواد بره به بانک و اطلاعات را چک کنه.

نویسنده: علی اکبر کورش فر
تاریخ: ۱۳۹۳/۰۵/۱۴ ۸:۳۴

بله. البته. ولی به نظر شما چه نیازی هستش وقتی که فیلد مربوطه پر شد دوباره به فیلد برگرده. این فقط در حالتی هستش که کاربر بخواد مقدار رو تغییر بده. پس اولویت داره استفاده از این کار در برابر استفاده نکردن

نویسنده: جوکار
تاریخ: ۱۳۹۳/۰۶/۰۱ ۱۵:۴۸

روش بسیار جالبی بود.اما یک مشکل که در این روش با آن روبرو میشویم این است که هنگام ویرایش یک رکورد موجود در بانک ، اگر قرار نباشد فیلد مورد نظر بروز رسانی شود ، اکشن متد ذکر شده دست ما را می‌بندد و اجازه آپدیت را نمی‌دهد.
شاید یک راه ایجاد یک viewModel جداگانه برای آپدیت باشد که در آن از remote attribute صرف نظر شود ، اما این راه حل زیاد جالب به نظر نمی‌رسد!
آیا راه حل مناسب‌تری وجود دارد؟

نویسنده: محسن خان
تاریخ: ۱۳۹۳/۰۶/۰۲ ۱۹:۶

برای حالت ویرایش AdditionFields آن کاربرد داره. مثلا فیلد Id رو اینجا میشه ارسال کرد تا مشخص باشه حالت ویرایش هست. در حالت ثبت معمولی، خوب هنوز Id رکورد مشخص نیست و نال هست.

نویسنده: علی خسروی
تاریخ: ۱۳۹۳/۰۷/۲۲ ۱۲:۵۷

با سلام و تشکر؛ در صورتی که قصد داشته باشیم هنگام remote یک loader هم نشون بدیم باید چکار کرد.

نویسنده: محسن خان
تاریخ: ۱۳۹۳/۰۷/۲۲ ۱۴:۱۲

از روال‌های رخدادگردان عمومی ajaxStart و ajaxComplete استفاده کنید.

نویسنده: م علی خسروی
تاریخ: ۱۳۹۳/۰۷/۲۳ ۹:۷

با تشکر
در واقع من سه input دارم که می‌خام remote بشند برای هر کدام هم یه loader در کنارش قرار دارم ، چه طوری میشه فهمید الان کدوم input در صفحه ajax رو start کرده تا loader اون نمایش داده بشه

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۷/۲۳ ۱۰:۴۷

اگر به فایل jquery.validate.js مراجعه کنید، در قسمت remote آن، متد startRequest پیش از شروع عملیات Ajax و متد

stopRequest پس از پایان کار فراخوانی می‌شوند.

```
prototype: {
startRequest: function( element ) {
  //...
},
stopRequest: function( element, valid ) {
  //...
},
```

این دو متد را باید برای نمایش loading بازنویسی کرد. برای مثال:

```
var originalStartRequest = $.validator.prototype.startRequest;
$.validator.prototype.startRequest = function (element) {
    // یافتن عنصر در حال بررسی
    var container = $('form').find("[data-valmsg-for='" + element.name + "']");
// افزودن کلاس نمایش منتظر بمانید
    container.addClass('loading');

// فراخوانی متد اصلی برای انجام کارهای درونی افزونه
    originalStartRequest.apply(this, arguments);
};

var originalStopRequest = $.validator.prototype.stopRequest;
$.validator.prototype.stopRequest = function (element) {
    // یافتن عنصر در حال بررسی
    var container = $('form').find("[data-valmsg-for='" + element.name + "']");
// حذف  کلاس نمایش منتظر بمانید
    container.removeClass('loading');

// فراخوانی متد اصلی برای انجام کارهای درونی افزونه
    originalStopRequest.apply(this, arguments);
};
```

در اینجا loading به span مخفی data-valmsg-for اضافه می‌شود.

```
<span class="field-validation-valid" data-valmsg-replace="true" data-valmsg-for="Url"></span>
```

نمونه‌ی این بازنویسی در مطلب « اعتبار سنجی سمت کاربر wysiwyg-editorها در ASP.NET MVC » هم انجام شده‌است.

نویسنده: حامد رشنو
تاریخ: ۱۳۹۳/۰۸/۳۰ ۱۹:۴۱

با سلام
من از این روش استفاده کردم خیلی جالبه، فقط یک مشکل ، توی سیستم خودم خوب جواب میده ولی وقتی سایت رو پابلیش میکنم و روی هاست میزارم کار نمیکنه. نمیدونم مشکل از کجاست.

نویسنده: محسن خان
تاریخ: ۱۳۹۳/۰۸/۳۰ ۲۰:۱۱

خطاهای ممکن را به این صورت بررسی کنید: نحوه استفاده از افزونه Firebug برای دیباگ برنامه‌های ASP.NET مبتنی بر jQuery

عنوان: **لغو اجرای یک اکشن فیلتر برای یک اکشن خاص در MVC**
نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۳/۱۱/۰۷ ۲۲:۵
آدرس: www.dotnettips.info
گروه‌ها: MVC, ASP.Net MVC, Attributes, ActionFilters

بسیار پیش می‌آید که یک کنترلر را به یک اکشن فیلتر خاص مزین کنیم. در این صورت تمامی اکشن‌های موجود در کنترلر مربوطه مجاب به اجرای اکشن فیلتر مورد نظر می‌شوند. اما بسیار پیش می‌آید که نخواهیم یک اکشن خاص در کنترلر مذکور اکشن فیلتر مورد نظر را اجرا کند.
یک راهکار ساده اما (به نظر شخصی من) غیر منطقی این است که تک تک اکشن هایی را که می‌خواهیم اکشن فیلتر مورد نظر را اجرا کنند، مزین کنیم و اکشن‌هایی که نمی‌خواهیم اکشن فیلتر مورد نظر را اجرا کنند به اکشن فیلتر مورد نظر مزین نمی‌کنیم. اما فرض کنید تعداد اکشن‌های ما زیاد باشند؛ به نظر این روش غیر منطقی و غیر بهینه است.

یکی از مشکلاتی که در یکی از پروژه‌ها حدود سه روز وقت من را گرفت همین کار بود. زمانی که از یک تصویر امنیتی جهت مقابله با ربوت‌های استفاده میکردم به این مشکل برخورد کردم. کنترلر من به یک اکشن فیلتر فشرده سازی محتوا مزین بود. در نتیجه اکشنی که تصویر امنیتی را تولید میکرد نیز اکشن فیلتر فشرده سازی را اجرا می‌کرد و پس از فشرده سازی باعث می‌شد که تصویر امنیتی نشان داده نشود، چون فرمت تصویری آن بهم ریخته بود. یک راهکار را که پس از جستجو به آن رسیدم، جهت استفاده‌ی دوستان مطرح می‌کنم.

برای اینکه از اجرای چنین اکشن فیلتر هایی جلوگیری کنیم نیاز است کمی اکشن فیلتر مورد نظر را دستکاری کنیم.
برای این کار باید مراحل زیر را انجام داد:
1- ابتدا یک Attribute خالی را تعریف می‌کنیم.
2- سپس اکشن فیلتر دلخواهی را تعریف کرده و در زمان اجرا بررسی می‌کنیم اگر متد (اکشن) مورد نظر با Attribute تعریفی در مرحله یک مزین شده بود، در نتیجه اکشن فیلتر را اجرا نمی‌کنیم.
3- هر اکشنی را که نمی‌خواهیم اکشن فیلتر تعریفی مرحله 2 را اجرا کند، آن را به Attribute مرحله یک مزین میکنیم.
به این ترتیب می‌توانیم از اجرای اکشن فیلتر دلخواه روی متدها یا اکشن‌های دلخواه جلوگیری کنیم. در ادامه نحوه‌ی تعریف آنها را در زیر مشاهده میکنید.

1- تعریف یک Attribute دلخواه مثلا با نام DisableCompression

```
[AttributeUsage(AttributeTargets.Class | AttributeTargets.Method, AllowMultiple = false, Inherited =
true)]
public sealed class DisableCompression : Attribute { }
```

2- تعریف اکشن فیلتر دلخواه مثلا با نام CompressionFilter

```
[AttributeUsage(AttributeTargets.Class | AttributeTargets.Method, Inherited = true, AllowMultiple =
false)]
public class CompressionFilter : ActionFilterAttribute
{
    public override void OnActionExecuting(ActionExecutingContext filterContext)
    {
        bool disabled = filterContext.ActionDescriptor.IsDefined(typeof(DisableCompression), true) ||

filterContext.ActionDescriptor.ControllerDescriptor.IsDefined(typeof(DisableCompression), true);
        if (disabled)
            return;

        // کدهای دلخواه اکشن فیلتر مورد نظر
    }
}
```

3- در این مرحله هر اکشنی را که نمی‌خواهیم اکشن فیلتر CompressionFilter را اجرا کند به Attribute با نام

DisableCompression مزین میکنیم.

```
[CompressionFilter]
public abstract class BaseController : Controller
{
}

public class SomeController : BaseController
{
    public ActionResult WantThisActionCompressed()
    {
        // code
    }

    [DisableCompression]
    public ActionResult DontWantThisActionCompressed()
    {
        // code
    }
}
```

با این کار اکشن WantThisActionCompressed اکشن فیلتر CompressionFilter را اجرا می‌کند اما اکشن DontWantThisActionCompressed چون مزین به DisableCompression شده‌است، پس در نتیجه اکشن فیلتر CompressionFilter بر روی آن اجرا نخواهد شد.