عنوان: آشنایی با Fluent Html Helpers در MVC
نویسنده: حمیدرضا صادقیان
تاریخ: ۱۳۹۲/۰۱/۳۰ ۱۸:۲۰
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net MVC, Fluent Html Helpers, MVC

همان طور که قبلا نیز اشاره شد اینجا به صورت خلاصه هدف Fluent API فراهم آوردن روشی است که بتوان متدها را زنجیر وار فراخوانی کرد و به این ترتیب خوانایی کد نوشته شده را بالا برد.

اما در این مقاله سعی شده تا کاربرد آن در یک برنامه MVC رو به صورت استفاده در helperها شرح دهیم.
در اینجا مثالی رو شرح میدهیم که در آن کنترل هایی از جنس input به صورت helper ساخته و برای فرستادن ویژگی‌های اچ تی ام ال (HTML Attributes) آن از Fluent Html Helpers بهره میگیریم بدیهی است که از این Fluent میتوان برای helperهای دیگر هم استفاده کرد.
در ابتدا یک نگاه کلی به کد ایجاد شده می‌اندازیم :

```
@Html.RenderInput(
        Attributes.Configure()
            .AddType("text")
            .AddId("UserId")
            .AddName("UserId")
            .AddCssClass("TxtBoxCssClass")
    )
```

که باعث ساخته شدن یک کنترل تکست باکس با مشخصات زیر در صفحه می‌شود ، همان طور که مشاهده می‌کنیم تمام ویژگی‌ها برای کنترل ساخته شده اند.

```
<input class="TxtBoxCssClass" id="UserId" name="UserId" type="text">
```

در ادامه به سراغ پیاده سازی کلاس Attributes برای این مثال می‌رویم ، این کلاس باید از کلاس RouteValueDictionary ارث بری داشته باشد، این کلاس یک دیکشنری برای ما آماده می‌کند تا مقادیرمون رو در آن بریزیم در اینجا برای ویژگی‌ها و مقادیرشون.

```
public class Attributes : RouteValueDictionary    {
        /// <summary>
        /// Configures this instance.
        /// </summary>
        /// <returns></returns>
        public static Attributes Configure()
        {
            return new Attributes();
        }

        /// <summary>
        /// Adds the type.
        /// </summary>
        /// <param name="value">The value.</param>
        public Attributes AddType(string value)
        {
            this.Add("type", value);
            return this;
        }

        /// <summary>
        /// Adds the name.
        /// </summary>
        /// <param name="value">The value.</param>
        public Attributes AddName(string value)
        {
            this.Add("name", value);
            return this;
        }

        /// <summary>
        /// Adds the id.
        /// </summary>
```

```
        /// <param name="value">The value.</param>
        public Attributes AddId(string value)
        {
            this.Add("id", value);
            return this;
        }

        /// <summary>
        /// Adds the value.
        /// </summary>
        /// <param name="value">The value.</param>
        public Attributes AddValue(string value)
        {
            this.Add("value", value);
            return this;
        }

        /// <summary>
        /// Adds the CSS class.
        /// </summary>
        /// <param name="value">The value.</param>
        public Attributes AddCssClass(string value)
        {
            this.Add("class", value);
            return this;
        }
    }
```

متد استاتیک Configure همیشه به عنوان شروع کننده fluent و از اون برای ساختن یک وهله جدید از کلاس Attributes استفاده میشه و بقیه متدها هم کار اضافه کردن مقادیر رو مثل یک دیکشنری انجام میدهند.

حالا به سراغ پیاده سازی helper extension مون میرویم

```
public static MvcHtmlString RenderInput(this HtmlHelper htmlHelper, Attributes attributes)
        {
            TagBuilder input = new TagBuilder("input");
            input.MergeAttributes(attributes);
            return new MvcHtmlString(input.ToString(TagRenderMode.SelfClosing));
        }
```

با استفاده از کلاس tagbuilder تگ input را ساخته و ویژگی‌های فرستاده شده به helper رو با اون ادغام می‌کنیم (با استفاده از MergeAttributes)

این یک مثال ساده بود از این کار امیدوارم مفید واقع شده باشد

مثال پروژه