عنوان: **استفاده از Razor در فایل‌های JavaScript و CSS**
نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۴/۰۲/۳۰ ۱۶:۳۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, MVC, CSS

یکی از مشکلات سینتکس Razor سمت سرور، این است که در فایل‌های JavaScript و CSS سمت کاربر نمی‌توانیم از آن استفاده کنیم، به عنوان مثال فرض کنید در یک فایل JavaScript نیاز به مشخص سازی آدرس یک اکشن متد دارید؛ مثلاً انجام یک عملیات ای‌جکسی. در این حالت به عنوان یک Best Practice بهتر است از Url.Action استفاده کنید. اما همانطور که عنوان شد، این امکان یعنی استفاده از سینتکس Razor در فایل‌های JS و CSS مهیا نیست.

ساده‌ترین راه‌حل، تولید ویوهای سمت سرور JavaScript ایی است. برای اینکار تنها کاری که باید انجام دهیم، تغییر مقدار Content-Type صفحه به مقدار موردنظر می‌باشد؛ مثلاً text/javascript برای فایل‌های JS و text/css برای فایل‌های CSS. به عنوان مثال برای فایل‌های CSS به این صورت عمل خواهیم کرد:

```
public ActionResult Style()
{
            Response.ContentType = "text/css";
            var model = new Style
            {
                Color = "red",
                Background = "blue"
            };
            return View(model);
}
```

برای ویوی آن نیز خواهیم داشت:

```
@model ExternalJavaScript.Models.Style
@{
    Layout = null;
}
body {
    color : @Model.Color;
    background-color : @Model.Background;
}
```

در نهایت ویوی فوق را به عنوان فایل CSS در فایل Layout استفاده خواهیم کرد:

```
<link rel="stylesheet" href="@Url.Action("Style","Home")" />
```

برای حالت فوق می‌توانیم یک اکشن فیلتر به صورت زیر تهیه کنیم:

```
public class ContentType : ActionFilterAttribute
    {
        private string _contentType;
        public ContentType(string ct)
        {
            this._contentType = ct;
        }

        public override void OnActionExecuted(ActionExecutedContext context) { /* nada */ }
        public override void OnActionExecuting(ActionExecutingContext context)
        {
            context.HttpContext.Response.ContentType = this._contentType;
        }
    }
```

و برای استفاده از آن خواهیم داشت:

```
[ContentType("text/css")]
public ActionResult Style()
 {
     var model = new Style
      {
                Color = "red",
                Background = "blue"
      };
     return View(model);
}
```

برای فایل‌های JS نیز می‌توانیم از یک View به عنوان محل قرارگیری کدهای جاوا اسکریپت استفاده کنیم:

```
public class JavaScriptSettingsController : Controller
{
        public ActionResult Index()
        {
            return PartialView();
        }
}
```

در این حالت در داخل فایل Index.cshtml کدهای جاوا اسکریپت را همراه با سینتکس Razor می‌توانیم بنویسیم:

```
$(function(){
    $.post('@Url.Action("GetData", "Home")', function (data) {
        $('.notificationList').html(data);
            if ($(data).filter("li").length != 0) {
                $('#notificationCounter').html($(data).filter("li").length);
            }
    });
});
```

سپس در داخل فایل Layout.cshtml_ می‌توانیم به ویوی فوق ارجاعی داشته باشیم:

```
<script src="/JavaScriptSettings"></script>
```

این روش به خوبی برای ویوهای JS و CSS کار خواهد کرد؛ اما از آنجائیکه ویوی ما توسط ویژوال استودیو به عنوان یک فایل JS و یا CSS معتبر شناخته نمی‌شود، Intellisense برای آن مهیا نیست. برای فعال سازی Intellisense و همچنین معتبر شناخته شدن ویوی فوق، بهترین راه‌حل قرار دادن کدهای JS درون بلاک script است (برای فایل‌های CSS نیز همینطور):

```
<script>
    $(function () {
        $.post('@Url.Action("Index", "Home")', function (data) {
            $('.notificationList').html(data);
            if ($(data).filter("li").length != 0) {
                $('#notificationCounter').html($(data).filter("li").length);
            }
        });
    });
</script>
```

اما با اجرای برنامه، در کنسول مرورگر بلافاصله خطای Uncaught SyntaxError: Unexpected token < را دریافت خواهید کرد. در این حالت به روشی نیاز داریم که در زمان اجرا بلاک script را حذف نمائید. بنابراین از یک اکشن فیلتر سفارشی برای اینکار استفاده خواهیم کرد. کار این اکشن فیلتر، تغییر مقدار Content-Type و همچنین حذف بلاک مورد نظر می‌باشد:

```
public class ExternalFileAttribute : ActionFilterAttribute
    {
        private readonly string _contentType;
        private readonly string _tag;
        public ExternalFileAttribute(string ct, string tag)
        {
            this._contentType = ct;
            _tag = tag;
```

```
        }

        public override void OnResultExecuted(ResultExecutedContext filterContext)
        {
            var response = filterContext.HttpContext.Response;
            response.Filter = new StripEnclosingTagsFilter(response.Filter, _tag);
            response.ContentType = _contentType;
        }

        private class StripEnclosingTagsFilter : MemoryStream
        {
            private static Regex _leadingOpeningScriptTag;
            private static Regex _trailingClosingScriptTag;

            //private static string Tag;

            private readonly StringBuilder _output;
            private readonly Stream _responseStream;

            /*static StripEnclosingTagsFilter()
            {
                LeadingOpeningScriptTag = new Regex(string.Format(@"^\s*<{0}[^>]*>", Tag),
RegexOptions.Compiled);
                TrailingClosingScriptTag = new Regex(string.Format(@"</{0}>\s*$", Tag),
RegexOptions.Compiled);
            }*/

            public StripEnclosingTagsFilter(Stream responseStream, string tag)
            {
                _leadingOpeningScriptTag = new Regex(string.Format(@"^\s*<{0}[^>]*>", tag),
RegexOptions.Compiled);
                _trailingClosingScriptTag = new Regex(string.Format(@"</{0}>\s*$", tag),
RegexOptions.Compiled);

                _responseStream = responseStream;
                _output = new StringBuilder();
            }

            public override void Write(byte[] buffer, int offset, int count)
            {
                string response = GetStringResponse(buffer, offset, count);
                _output.Append(response);
            }

            public override void Flush()
            {
                string response = _output.ToString();

                if (_leadingOpeningScriptTag.IsMatch(response) &&
_trailingClosingScriptTag.IsMatch(response))
                {
                    response = _leadingOpeningScriptTag.Replace(response, string.Empty);
                    response = _trailingClosingScriptTag.Replace(response, string.Empty);
                }

                WriteStringResponse(response);
                _output.Clear();
            }

            private static string GetStringResponse(byte[] buffer, int offset, int count)
            {
                byte[] responseData = new byte[count];
                Buffer.BlockCopy(buffer, offset, responseData, 0, count);

                return Encoding.Default.GetString(responseData);
            }

            private void WriteStringResponse(string response)
            {
                byte[] outdata = Encoding.Default.GetBytes(response);
                _responseStream.Write(outdata, 0, outdata.GetLength(0));
            }
        }
    }
```

در نهایت می‌توانیم اکشن‌متد موردنظرمان را با فیلتر سفارشی مزین کنیم:

```
[ExternalFile("text/javascript", "script")]
```

```
public ActionResult Index()
{
      return PartialView();
}
```

برای تولید ویوهای CSS نیز کافی است مقادیر فیلتر را تغییر دهیم:

```
[ExternalFile("text/css", "style")]
public ActionResult Style()
{
            var model = new Style
            {
                Color = "red",
                Background = "blue"
            };
            return View(model);
}
```

نظرات خوانندگان

نویسنده: حسین شجاعی
تاریخ: ۱۳۹۴/۰۲/۳۰ ۲۳:۴

سپاس از مطلبتون.
فقط یک سوال : این پاسخ صحیحه؟
موفق باشید

نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۴/۰۲/۳۱ ۰:۴۸

سوال مربوط به استفاده از razor درون ویو (در واقع درون تگ اسکریپت) است، مطلب جاری درباره استفاده از razor درون یک فایل جاوا اسکریپت و یا CSS اکسترنال می‌باشد.

نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۴/۰۲/۳۱ ۱۹:۲۸

برای پیاده سازی minfication و bundling روشی هست؟

نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۴/۰۳/۰۱ ۱۳:۲۲

برای bundle کردن من روشی به ذهنم نمی‌رسه ولی برای minification می‌تونید از روش حذف فضاهای خالی در خروجی صفحات ASP.NET MVC استفاده کنید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۳/۰۲ ۲۲:۵۳

یک روش دیگر هم استفاده از ویژگی‌های *-data مربوط به HTML 5 است. برای مثال اگر صرفا هدف مشخص سازی Url و یا اطلاعاتی از این دست است، بهتر است این موارد را داخل فایل اسکریپت قرار نداد. در همان View معمولی یک ویژگی data سفارشی را ایجاد کنید:

```
<div data-url="@Url.Action(....)">
</div>
```

و بعد در فایل اسکریپت خارجی به این نحو قابل خواندن خواهد بود:

```
var url = $("div").data("url") ;
```