عنوان: **ASP.NET MVC #7**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۱/۱۲ ۱۰:۵۳:۰۰
آدرس: www.dotnettips.info
گروه‌ها: MVC

آشنایی با Razor Views

قبل از اینکه بحث جاری ASP.NET MVC را بتوانیم ادامه دهیم و مثلا مباحث دریافت اطلاعات از کاربر، کار با فرم‌ها و امثال آن‌را بررسی کنیم، نیاز است حداقل به دستور زبان یکی از View Engineهای ASP.NET MVC آشنا باشیم.
MVC3 موتور View جدیدی را به نام Razor معرفی کرده است که به عنوان روش برگزیده ایجاد Viewها در این سیستم به شمار می‌رود و فوق العاده نسبت به ASPX view engine سابق، زیباتر، ساده‌تر و فشرده‌تر طراحی شده است و یکی از اهداف آن تلفیق code و markup می‌باشد. در این حالت دیگر پسوند فایل‌های Viewها همانند سابق ASPX نخواهد بود و به cshtml و یا vbhtml تغییر یافته است. همچنین برخلاف web forms view engine از System.Web.Page مشتق نشده است. و باید دقت داشت که Razor یک زبان برنامه نویسی جدید نیست. در اینجا از مخلوط زبان‌های سی شارپ و یا ویژوال بیسیک به همراه تگ‌های html استفاده می‌شود.
البته این را هم باید عنوان کرد که این مسایل سلیقه‌ای است. اگر با web forms view engine راحت هستید، با همان کار کنید. اگر با هیچکدام از این‌ها راحت نیستید (!) نمونه‌های دیگر هم وجود دارند ، مثلا:

Spark
NHaml
SharpDOM
SharpTiles
Wing Beats
string-template-view-engine-mvc
Bellevue
Brail
Hasic
NDjango

Razor Views یک سری قابلیت جالب را هم به همراه دارند:
1) امکان کامپایل آن‌ها به درون یک DLL وجود دارد. مزیت: استفاده مجدد از کد، عدم نیاز به وجود صریح فایل cshtml یا vbhtml بر روی دیسک سخت.
2) آزمون پذیری: از آنجائیکه Razor viewها به صورت یک کلاس کامپایل می‌شوند و همچنین از System.Web.Page مشتق نخواهند شد، امکان بررسی HTML نهایی تولیدی آن‌هابدون نیاز به راه اندازی یک وب سرور وجود دارد.
3) IntelliSense ویژوال استودیو به خوبی آن‌را پوشش می‌دهد.
4) با توجه به مواردی که ذکر شد، یک اتفاق جالب هم رخ داده است: امکان استفاده از Razor engine خارج از ASP.NET MVC هم وجود دارد. برای مثال یک سرویس ویندوز NT طراحی کرده‌اید که قرار است ایمیل فرمت شده‌ای به همراه اطلاعات مدل‌های شما را در فواصل زمانی مشخص ارسال کند؟ می‌توانید برای طراحی آن از Razor engine استفاده کنید و تهیه خروجی نهایی HTML آن نیازی به راه اندازی وب سرور و وهله سازی HttpContext ندارد.

ساختار پروژه مثال جاری

در ادامه مرور سریعی خواهیم داشت بر دستور زبان Razor engine و جهت نمایش این قابلیت‌ها، یک مثال ساده را در ابتدا با مشخصات زیر ایجاد خواهیم کرد:
الف) یک ASP.NET MVC 3 project empty را ایجاد کنید و نوع View engine را هم در ابتدای کار Razor انتخاب نمائید.

ب) دو کلاس زیر را به پوشه مدل‌های برنامه اضافه کنید:

```
namespace MvcApplication3.Models
{
    public class Product
    {
        public Product(string productNumber, string name, decimal price)
        {
            Name = name;
            Price = price;
            ProductNumber = productNumber;
        }
        public string ProductNumber { get; set; }
        public string Name { get; set; }
        public decimal Price { get; set; }
    }
}
```

```
using System.Collections.Generic;

namespace MvcApplication3.Models
{
    public class Products : List<Product>
    {
        public Products()
        {
            this.Add(new Product("D123", "Super Fast Bike", 1000M));
            this.Add(new Product("A356", "Durable Helmet", 123.45M));
            this.Add(new Product("M924", "Soft Bike Seat", 34.99M));
        }
    }
}
```

کلاس Products صرفا یک منبع داده تشکیل شده در حافظه است. بدیهی است هر نوع ORM ایی که یک ToList را بتواند در اختیار شما قرار دهد، توانایی تشکیل لیست جنریکی از محصولات را نیز خواهد داشت و تفاوتی نمی‌کند که کدامیک مورد استفاده قرار گیرد.

ج) سپس یک کنترلر جدید به نام ProductsController را به پوشه Controllers برنامه اضافه می‌کنیم:

```
using System.Web.Mvc;
using MvcApplication3.Models;

namespace MvcApplication3.Controllers
{
    public class ProductsController : Controller
    {
        public ActionResult Index()
        {
            var products = new Products();
            return View(products);
        }
    }
}
```

د) بر روی نام متد Index کلیک راست کرده، گزینه Add view را جهت افزودن View متناظر آن، انتخاب کنید. البته می‌شود همانند قسمت پنجم گزینه Create a strongly typed view را انتخاب کرد و سپس Product را به عنوان کلاس مدل انتخاب نمود و در آخر خیلی سریع یک لیست از محصولات را نمایش داد، اما فعلا از این قسمت صرفنظر نمائید، چون می‌خواهیم آن را دستی ایجاد کرده و توضیحات و نکات بیشتری را بررسی کنیم.

ه) برای اینکه حین اجرای برنامه در VS.NET هربار نخواهیم که آدرس کنترلر Products را دستی در مرورگر وارد کنیم، فایل

Global.asax.cs را گشوده و سپس در متد RegisterRoutes، در سطر Parameter defaults، مقدار پیش فرض کنترلر را مساوی Products قرار دهید.

مرجع سریع Razor

ابتدا کدهای View متد Index را به شکل زیر وارد نمائید:

```
@model List<MvcApplication3.Models.Product>
@{
    ViewBag.Title = "Index";
    var number = 12;
    var data = "some text...";
    <h2>line1: @data</h2>

    @:line-2: @data <br />
    <text>line-3:</text> @data
}
<br />
site@(data)
<br />
@@name
<br />
@(number/10)
<br />
First product: @Model.First().Name
<br />
@if (@number>10)
{
  <span>@data</span>
}
else
{
  <text>Plain Text</text>
}
<br />
@foreach (var item in Model)
{
    <li>@item.Name, $@item.Price </li>
}

@*
    A Razor Comment
*@

<br />
@("First product: " + Model.First().Name)
<br />
<img src="@(number).jpg" />
```

در ادامه توضیحات مرتبط با این کدها ارائه خواهد شد:

1) نحوه معرفی یک قطعه کد

```
@model List<MvcApplication3.Models.Product>
@{
    ViewBag.Title = "Index";
    var number = 12;
    var data = "some text...";
    <h2>line1: @data</h2>

    @:line-2: @data <br />
    <text>line-3:</text> @data
}
```

این کدها متعلق به Viewایی است که در قسمت (د) بررسی ساختار پروژه مثال جاری، ایجاد کردیم. در ابتدای آن هم نوع model

مشخص شده تا بتوان ساده‌تر به اطلاعات شیء Model به کمک IntelliSense دسترسی داشت.
برای ایجاد یک قطعه کد در Viewایی از نوع Razor به این نحو عمل می‌شود:

```
@{  ...Code Block.... }
```

در اینجا مجاز هستیم کدهای سی شارپ را وارد کنیم. یک نکته جالب را هم باید درنظر داشت: امکان نوشتن تگ‌های html هم در این بین وجود دارد (بدون اینکه مجبور باشیم قطعه کد شروع شده را خاتمه دهیم، به حالت html معمولی سوئیچ کرده و دوباره یک قطعه کد دیگر را شروع نمائیم). مانند line1 مثال فوق. اگر کمی پایین‌تر از این سطر مثلا بنویسیم line2 (به عنوان یک برچسب) کامپایلر ایراد خواهد گرفت، زیرا این مورد نه متغیر است و نه از پیش تعریف شده است. به عبارتی نباید فراموش کنیم که اینجا قرار است کد نوشته شود. برای رفع این مشکل دو راه حل وجود دارد که در سطرهای دو و سه ملاحظه می‌کنید. یا باید از تگی به نام text برای معرفی یک برچسب در این میان استفاده کرد (سطر سه) یا اگر قرار است اطلاعاتی به شکل یک متن معمولی پردازش شود ابتدای آن مانند سطر دوم باید یک @: قرار گیرد.
کمی پایین‌تر از قطعه کد معرفی شده در بالا بنویسید:

```
<br />
site@data
```

اکنون اگر خروجی این View را در مرورگر بررسی کنید، دقیقا همین site@data خواهد بود. چون در این حالت Razor تصور خواهد کرد که قصد داشته‌اید یک آدرس ایمیل را وارد کنید. برای این حالت خاص باید نوشت:

```
<br />
site@(data)
```

به این ترتیب data از متغیر data تعریف شده در code block قبلی برنامه دریافت و نمایش داده خواهد شد.
شبیه به همین حالت در مثال زیر هم وجود دارد:

```
<img src="@(number).jpg" />
```

در اینجا اگر پرانتزها را حذف کنیم، Razor فرض را بر این خواهد گذاشت که شیء number دارای خاصیت jpg است. بنابراین باید به نحو صریحی، بازه کاری را مشخص نمائیم.

بکار گیری این علامت @ یک نکته جنبی دیگر را هم به همراه دارد. فرض کنید در صفحه قصد دارید آدرس توئیتری شخصی را وارد کنید. مثلا:

```
<br />
@name
```

در این حالت View کامپایل نخواهد شد و Razor تصور خواهد کرد که قرار است اطلاعات متغیری به نام name را نمایش دهید.
برای نمایش این اطلاعات به همین شکل، یک @ دیگر به ابتدای سطر اضافه کنید:

```
<br />
@@name
```

2) نحوه معرفی عبارات

عبارات پس از علامت @ معرفی می‌شوند و به صورت پیش فرض Html Encoded هستند (در قسمت 5 در اینباره بیشتر توضیح داده شد):

```
First product: @Model.First().Name
```

در این مثال با توجه به اینکه نوع مدل در ابتدای View مشخص شده است، شیء Model به لیستی از Products اشاره می‌کند.

یک نکته:
مشخص سازی حد و مرز صریح یک متغیر در مثال زیر نیز کاربرد دارد:

```
<br />
@number/10
```

اگر خروجی این مثال را بررسی کنید مساوی 10/12 خواهد بود و محاسبه‌ای انجام نخواهد شد. برای حل این مشکل باز هم از پرانتز می‌توان کمک گرفت:

```
<br />
@(number/10)
```

3) نحوه معرفی عبارات شرطی

```
@if (@number>10)
{
  <span>@data</span>
}
else
{
  <text>Plain Text</text>
}
```

یک عبارت شرطی در اینجا با @if شروع می‌شود و سپس نکاتی که در «نحوه معرفی یک قطعه کد» بیان شد، در مورد عبارات داخل {} صادق خواهد بود. یعنی در اینجا نیز می‌توان عبارات سی شارپ مخلوط با تگ‌های html را نوشت.
یک نکته: عبارت شرطی زیر نادرست است. حتما باید سطرهای کدهای سی شارپ بین {} محصور شوند؛ حتی اگر یک سطر باشند:

```
@if( i < 1 ) int myVar=0;
```

4) نحوه استفاده از حلقه foreach

```
@foreach (var item in Model)
{
    <li>@item.Name, $@item.Price </li>
}
```

حلقه foreach نیز مانند عبارات شرطی با یک @ شروع شده و داخل {} بدنه آن نکات «نحوه معرفی یک قطعه کد» برقرار هستند (امکان تلفیق code و markup با هم).
کسانی که پیشتر با web forms کار کرده باشند، احتمالا الان خواهند گفت که این یک پس رفت است و بازگشت به دوران ASP کلاسیک دهه نود! ما به ندرت داخل صفحات aspx وب فرم‌ها کد می‌نوشتیم. مثلا پیشتر یک GridView وجود داشت و یک دیتاسورس که به آن متصل می‌شد؛ مابقی خودکار بود و ما هیچ وقت حلقه‌ای ننوشتیم. در اینجا هم این مساله با نوشتن برای مثال «html helpers» قابل کنترل است که در قسمت‌های بعدی به آن پرداخته خواهد شد. به عبارتی قرار نیست به این نحو با Viewهای Razor رفتار کنیم. این قسمت فقط یک آشنایی کلی با Syntax است.

5) امکان تعریف فضای نام در ابتدای View

```
@using namespace;
```

6) نحوه نوشتن توضیحات سمت سرور:

```
@*
    A Razor Comment / Server side Comment
*@
```

7) نحوه معرفی عبارات چند جزئی:

```
@("First product: " + Model.First().Name)
```

همانطور که ملاحظه می‌کنید، ذکر یک پرانتز برای معرفی عبارات چندجزئی کفایت می‌کند.

استفاده از موتور Razor خارج از ASP.NET MVC

پیشتر مطلبی را در مورد « تهیه قالب برای ایمیل‌های ارسالی یک برنامه ASP.Net » در این سایت مطالعه کرده‌اید. اولین سؤالی هم که در ذیل آن مطلب مطرح شده این است: «در برنامه‌های ویندوز چطور؟» پاسخ این است که کل آن مثال بر مبنای HttpContext.Current.Server.Execute کار می‌کند. یعنی باید مراحل وهله سازی HttpContext و شیء Server توسط یک وب سرور و درخواست رسیده طی شود و ... شبیه سازی آن آنچنان مرسوم و کار ساده‌ای نیست.
اما این مشکل با Razor وجود ندارد. به عبارتی در اینجا برای رندر کردن یک Razor View به html نهایی، نیازی به HttpContext نیست. بنابراین از این امکانات مثلا در یک سرویس ویندوز ان تی یا یک برنامه کنسول، WPF، WinForms و غیره هم می‌توان استفاده کرد.
برای اینکه بتوان از Razor خارج از ASP.NET MVC استفاده کرد، نیاز به اندکی کدنویسی هست مثلا استفاده از کامپایلر سی شارپ یا وی بی و کامپایل پویای کد و یک سری ست آپ دیگر. پروژه‌ای به نام RazorEngine این کپسوله سازی رو انجام داده و از اینجا http://razorengine.codeplex.com/ قابل دریافت است.

نظرات خوانندگان

نویسنده: Mohsen
تاریخ: ۱۳۹۱/۰۱/۱۲ ۲۳:۵۷:۳۰

لطفا کمی در مورد @model نوشته شده در سطر اول بیشتر توضیح بدید و اینکه چرا با حرف کوچک شروع شده.(آیا میشود به بیش از یک مدل در یک ویو اشاره کند؟)

نویسنده: Mohsen
تاریخ: ۱۳۹۱/۰۱/۱۳ ۰۰:۰۹:۳۲

یادم هست که در ASP Classic یک جاهایی مثل هنگام تعریف زبان اسکریپت نویسی سمت سرور از "@" استفاده می کردیم(انهم داخل %>).ولی مابقی جاها همان <% %> ولی مابین اینها نمی شد از Html مستقیما استفاده کرد.در واقع انعطاف پذیری بیشتری الان وجود داره...

ولی مهندس نصیری این @ ها آدم رو یاد $ های PHP می اندازه واقعا.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۱/۱۳ ۰۰:۱۱:۱۵

برای توضیحات بیشتر در مورد Model و model به قسمت پنجم مراجعه کنید.
توضیح تکمیلی:
- کلاس پایه‌ای در ASP.NET MVC وجود دارد به نام (^) WebViewPage. این کلاس حاوی تعاریف اولیه TempData، ViewBag، ViewData و ... Model است. این Model ریشه‌اش به اینجا بر می‌گردد و با حرف بزرگ شروع شده است. بنابراین در Viewهای سی شارپ Razor برای تعریف نوع مدل نیاز است بین model و شیء Model تفاوت وجود داشته باشد.
- در یک View شما هر تعداد مدل رو می‌تونید از طریق ViewBag و ViewData و غیره که در قسمت 5 توضیح داده شده، دریافت کنید و محدودیتی ندارد. اما این‌ها هیچکدام به معنای Strongly typed بودن View نیست. بنابراین برای حالت داشتن View از نوع Strongly typed، یکبار باید این نوع، تعریف شود.
البته یک راه هوشمندانه برای ارسال بیش از یک شیء به Model وجود دارد. یک کلاس تعریف کنید که خواص آن چندین شیء مورد نظر شما باشند. سپس این کلاس را به عنوان نوع Model در ابتدای View معرفی کنید. در اینجا به راحتی و به صورت Strongly typed با چند شیء می‌شود به عنوان Model کار کرد.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۱/۱۳ ۰۰:۳۰:۱۲

در PHP نمیشه یک code block رو با $ شروع کرد.

نویسنده: محمد صاحب
تاریخ: ۱۳۹۱/۰۱/۱۴ ۱۴:۴۲:۰۳

با تشکر از شما.
آقای نصیری این راه حل هوشمندانه در واقع همون ViewModel هست؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۱/۱۴ ۱۵:۲۵:۰۰

به عبارتی.

نویسنده: Salehi

تاریخ: ۱۳۹۱/۰۱/۱۶ ۲۳:۵۹:۳۱

به نظر میاد اکراه! دارید که اسمش رو viewModel بذارید. به خاطر تفاوتهاش با viewmodel در سیلورلایته؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۱/۱۷ ۱۲:۲۲:۲۴

واقعیت این است که پوشه Model در ASP.NET MVC باید به ViewModel از ابتدای کار تغییر نام پیدا می‌کرد. تمام مدل‌هایی که در اینجا از آن صحبت می‌کنیم ViewModel هستند. البته این نیاز به توضیح بیشتری دارد که در جای دیگری عرض خواهم کرد.

نویسنده: Salehi
تاریخ: ۱۳۹۱/۰۱/۱۸ ۲۲:۱۱:۲۴

الان که شما گفتید و من فکر کردم ،به نظرم همینطوره. البته واقعا جای توضیحات بیشتر داره که بیصبرانه منتظرش میمونم.
راستش اون موقع که شما آموزش سیلورلایت رو در برنامه داشتید، من اصلا نمی تونستم با MVVM ارتباط برقرار کنم و آخرش هم برام جا نیفتاد.ولی mvc با اینکه اولش برام خیلی سخت بود، ولی الان خیلی بهتره و با "قرارداد"هاش تا حد زیادی تونستم کنار بیام . به خاطر همین هم دنبال راهی از mvc به MVVM هستم!

نویسنده: Mojtaba
تاریخ: ۱۳۹۱/۰۲/۰۱ ۱۹:۱۴:۲۸

سلام
آیا برای محیط طراحی liveview برای razor وجود دارد

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۰۱ ۲۰:۳۶:۵۳

خیر. علت هم این است که صفحات پویای وب زمانی رندر می‌شوند که یک سری مراحل را پشت سر بگذارند و متفاوتند با صفحات ثابت HTML که در همان لحظه می‌شود خروجی را دید.
نیاز است وب سروری وجود داشته باشد، درخواستی به سرور ارسال شود، در ادامه کنترلر اطلاعاتی را از بانک اطلاعاتی دریافت کند و سپس به View ارسال کند. بنابراین این View در مرحله آخر سیکل قرار می‌گیرد و تمام این‌ها با هم یک سیستم را تشکیل می‌دهند. بنابراین جدا کردن یک قسمت از کل سیستم آنچنان معنایی ندارد.

نویسنده: امیرحسین م
تاریخ: ۱۳۹۱/۰۴/۲۸ ۱۵:۰

سلام آقای نصیری
کدتون یه مقداری اشکال داره
خط اول

```
@model List<MvcApplication3.Models.Product>
```

باید به

```
@model List<MvcApplication3.Models.Products>
```

تغییر کنه

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۲۸ ۱۶:۴

خیر. Products در اینجا خودش یک List است (به علت ارث بری صورت گرفته):

```
public class Products : List<Product>
```

بنابراین نیازی نیست که لیست یک لیست رو به عنوان مدل تعریف کرد.

نویسنده: احمد احمدی
تاریخ: ۱۳۹۱/۰۶/۲۹ ۱۹:۵۰

به مقدار رندر شده‌ی کد زیر توجه کنید :

```
@{
    string htmlContent = "&";
}
<span title="@Html.Raw(htmlContent)" id="@Html.Raw(htmlContent)">@Html.Raw(htmlContent)</span>
```

مقدار رندر شده :

```
<span title="&amp;" id="&amp;">&</span>
```

چرا مقادیر Attributeها encode شده اما مقدار بین تگ encode نشده است؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۶/۲۹ ۲۰:۴۶

اینجا

نویسنده: مصطفی
تاریخ: ۱۳۹۲/۱۰/۱۸ ۲۲:۱۵

سلام؛ در حلقه foreach نمی‌تونم به خاصیت‌ها دسترسی داشته باشم.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۰/۱۹ ۰:۵۲

- Model ذکر شده در حلقه با M بزرگ است و نه کوچک.
- ذکر @model با m کوچک برای تعریف Model در ابتدای View ضروری است.
- پس از تعریف کلاس‌های مدل برنامه، کامپایل را فراموش نکنید (خیلی از قسمت‌ها بر اساس Reflection کار می‌کنند و پس از کامپایل قابل دسترسی می‌شوند).

نویسنده: مهرداد پاک دل
تاریخ: ۱۳۹۲/۱۱/۰۷ ۱۱:۴۵

سلام؛ در سایت جاری زمانی که کاربر شناخته شده نیست منوهای بالا یک سری ایتم را نمایش می‌دهد و به محض وارد شدن کاربر منوها تغییر خواهند کرد . در mvc با کدام مکانیزم این کار را انجام می‌دهید . من منوه‌های پروژه خودم را با actionlink شبیه سازی کردم . تا اونجایی که من تحقیق کردم در web formsها از مولفه login view استفاده خواهد شد

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۰۷ ۱۴:۱۶

مباحث اعتبارسنجی در MVC را مطالعه کنید. پایه آن همین مطالب است. پس از اعتبارسنجی کاربر، در یک View می‌شود ساده if و

else نوشت. مثلا:

```
@if (User.Identity.IsAuthenticated && User.IsInRole("Administrator"))
{
     <div id="sidebar">
      data
     </div>
}
```

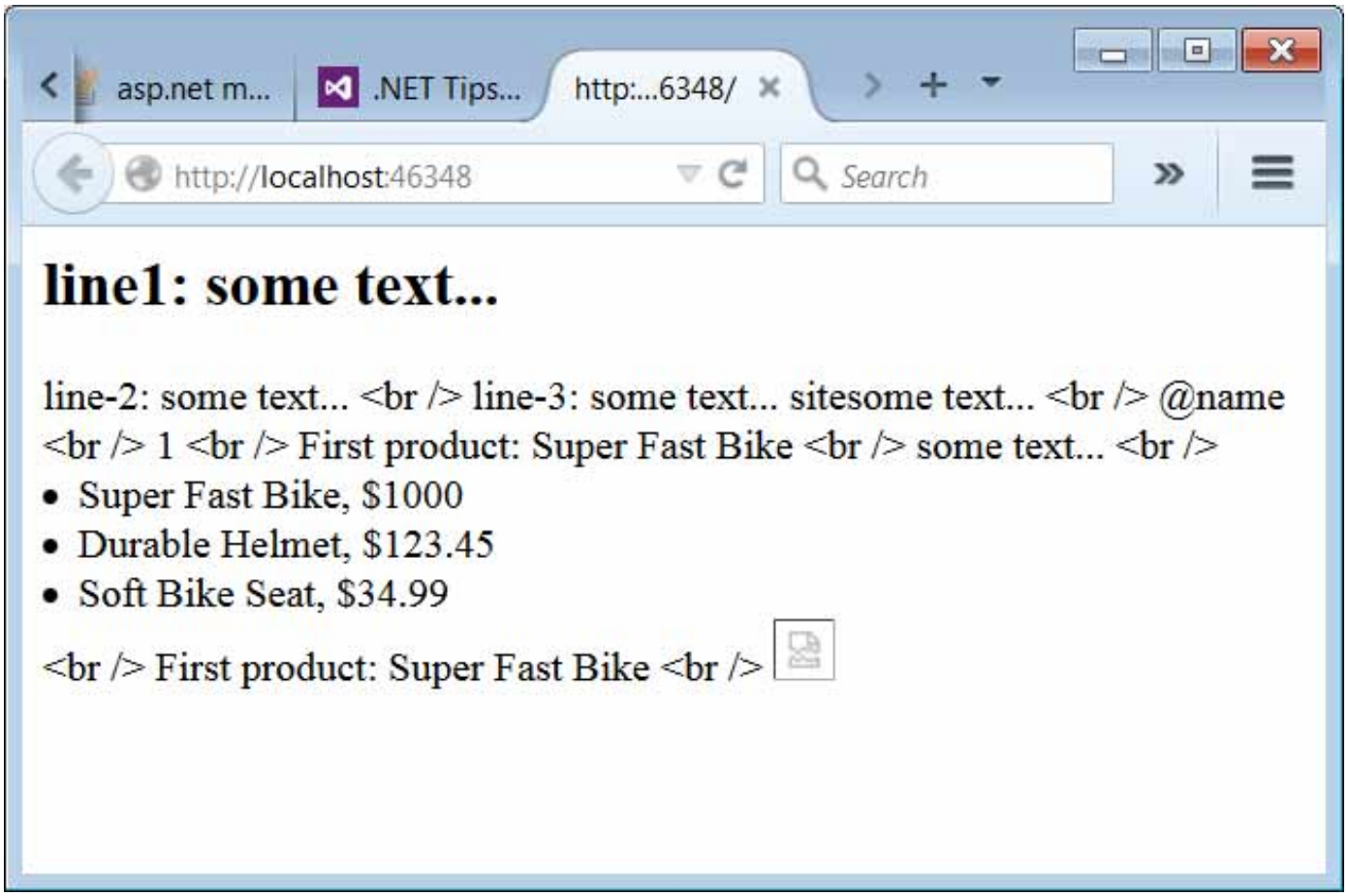
در این حالت اگر کاربر به سیستم لاگین کرده باشد و همچنین نقش Administrator نیز به او پیشتر انتساب داده شده باشد، اطلاعات خاصی را مشاهده خواهد کرد.

نویسنده: میثاق ابراهیمی
تاریخ: ۱۳۹۳/۱۰/۲۲ ۱۴:۶

کدهای view متد index را همانند بالا وارد کردم ولی به تعداد تگ‌های
 به من وارنینگ میدهد:

.Unknown element 'br' or element cannot be placed here

و در مرورگر تگهای br را همانطور نشان میدهد:



نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۱۰/۲۲ ۱۴:۳۴

- دریافت تمام مثال‌های MVC این سری (برای اینکه با copy/paste مثال‌ها مشکلی نداشته باشید): MVC_Samples
+ در متن ذکر شده: «عباراتی که پس از علامت @ معرفی می‌شوند، به صورت پیش فرض Html Encoded هستند (در قسمت 5

در اینباره بیشتر توضیح داده شد)». در قسمت پنجم « یک نکته امنیتی » و متد Html.Raw را مطالعه کنید.