عنوان: ASP.NET MVC #6
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۱/۱۱ ۰۰:۰۸:۰۰
آدرس: www.dotnettips.info
گروه‌ها: MVC

آشنایی با انواع ActionResult

در قسمت چهارم، اولین متد یا اکشنی که به صورت خودکار توسط VS.NET به برنامه اضافه شد، اینچنین بود:

```
using System.Web.Mvc;

namespace MvcApplication1.Controllers
{
    public class HomeController : Controller
    {
        //
        // GET: /Home/
        public ActionResult Index()
        {
            return View();
        }
    }
}
```

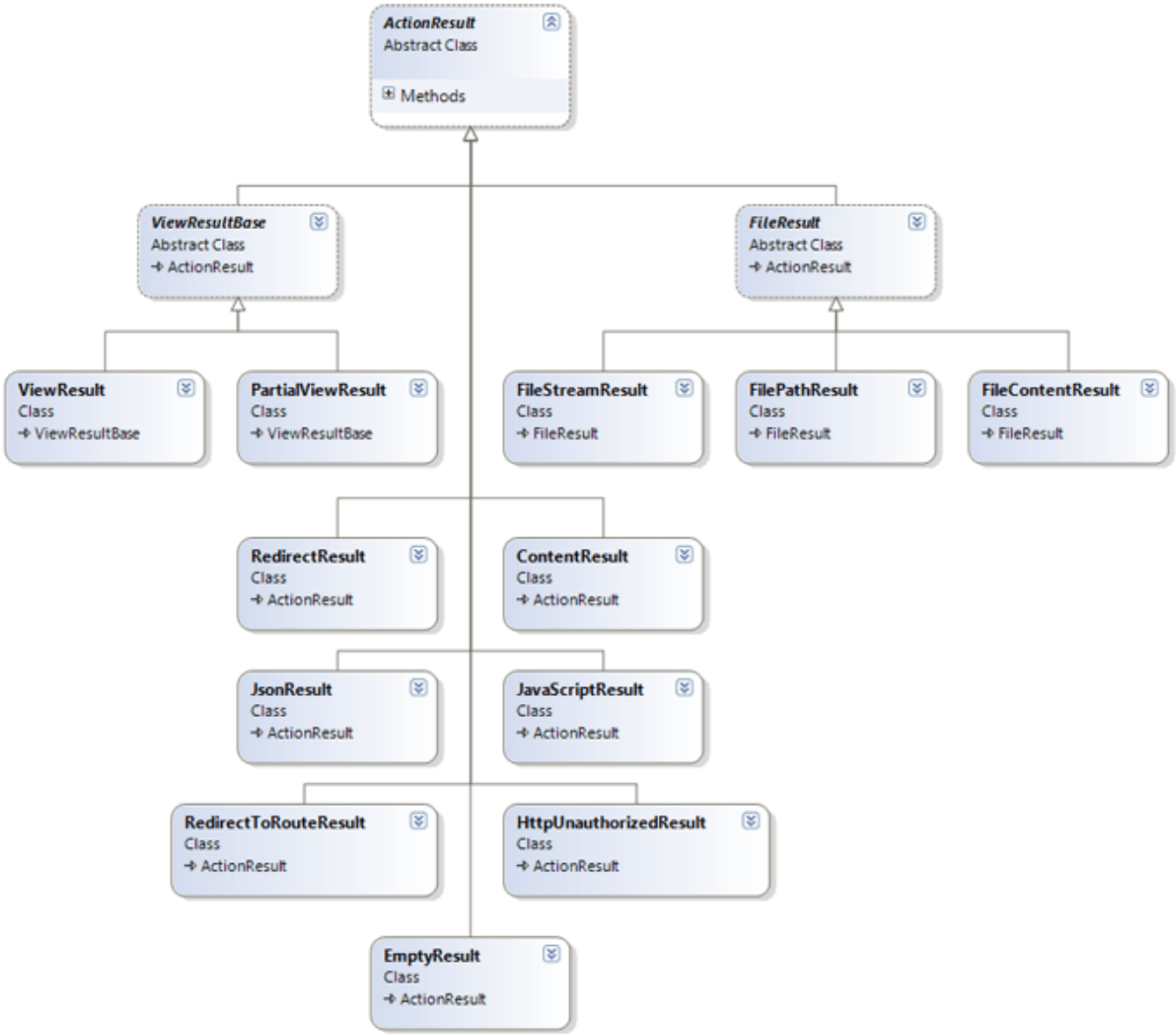
توضیحات تکمیلی مرتبط با خروجی از نوع ActionResult ایی را که مشاهده می‌کنید، در این قسمت ارائه خواهد شد.
رفتار یک کنترلر توسط متدهایی که در آن کلاس تعریف می‌شوند، مشخص می‌گردد. هر متد هم از طریق یک URL مجزا قابل دسترسی و فراخوانی خواهد بود. این متدها که به آن‌ها اکشن نیز گفته می‌شود باید عمومی بوده، استاتیک یا متد الحاقی (extension method) نباشند و همچنین دارای پارامترهایی از نوع ref و out نیز نباشند.
هر درخواست رسیده، به یک کنترلر و متدی عمومی در آن توسط سیستم مسیریابی، نگاشت خواهد شد. اگر علاقمند باشید که در یک کلاس کنترلر، متدی عمومی را از این سیستم خارج کنید، تنها کافی است آن‌را با ویژگی (attribute) به نام NonAction مزین کنید:

```
using System.Web.Mvc;

namespace MvcApplication2.Controllers
{
    public class HomeController : Controller
    {
        [NonAction]
        public string ShowData()
        {
            return "Text";
        }

        public ActionResult Index()
        {
            ViewBag.Message = string.Format("{0}/{1}/{2}",
                                            RouteData.Values["controller"],
                                            RouteData.Values["action"],
                                            RouteData.Values["id"]);
            return View();
        }

        public ActionResult Search(string data = "*")
        {
            // do something ...
            return View();
        }
    }
}
```

چند نکته در این مثال قابل ذکر است:
الف) در اینجا اگر شخصی آدرس http://localhost/home/showdata را درخواست نماید، با توجه به استفاده از ویژگی NonAction، با پیغام یافت نشد یا 404 مواجه می‌گردد.
ب) صرفنظر از پارامترهای یک متد و ساختار کلاس جاری، اطلاعات مسیریابی از طریق شیء RouteData.Values نیز در دسترس هستند که نمونه‌ای از آن‌را در اینجا بر اساس مقادیر پیش فرض تعاریف مسیریابی یک پروژه ASP.NET MVC ملاحظه می‌نمائید.
ج) در متد Search، از قابلیت امکان تعریف مقداری پیش فرض جهت آرگومان‌ها در سی شارپ 4 استفاده شده است. به این ترتیب اگر شخصی آدرس http://localhost/home/search را وارد کند، چون پارامتری را ذکر نکرده است، به صورت خودکار از مقدار پیش فرض آرگومان data استفاده می‌گردد.

انواع Action Results در ASP.NET MVC

در ASP.NET MVC بجای استفاده مستقیم از شیء Response، از شیء ActionResult جهت ارائه خروجی یک متد استفاده می‌شود و مهم‌ترین دلیل آن هم مشکل بودن نوشتن آزمون‌های واحد برای شیء Response است که وهله سازی آن مساوی است با به کار اندازی موتور ASP.NET و Http Runtime آن توسط یک وب سرور (بنابراین در ASP.NET MVC سعی کنید شیء Response را فراموش کنید).
سلسه مراتب ActionResultهای قابل استفاده در ASP.NET در تصویر زیر مشخص شده‌اند:



و در مثال زیر تقریبا انواع و اقسام ActionResultهای مهم و کاربردی ASP.NET MVC را می‌توانید مشاهده کنید:

```
using System.Web.Mvc;

namespace MvcApplication2.Controllers
{
    public class ActionResultsController : Controller
    {
        //http://localhost/actionresults/welcome
        public string Welcome()
        {
            return "Hello, World";
        }

        //http://localhost/actionresults/index
        public ActionResult Index() // or ContentResult
        {
            return Content("Hello, World");
        }

        //http://localhost/actionresults/SendMail
        public void SendMail()
```

```
        {
        }

        public ActionResult SendMailCompleted() // or EmptyResult
        {
            // do whatever
            return new EmptyResult();
        }

        public ActionResult GetFile() // or FilePathResult
        {
            return File(Server.MapPath("~/content/site.css"), "text/css", "mySite.css");
        }

        public ActionResult UnauthorizedStatus() // or HttpStatusCodeResult/HttpUnauthorizedResult
        {
            return new HttpUnauthorizedResult("You need to login first.");
        }

        public ActionResult Status() // or HttpStatusCodeResult
        {
            return new HttpStatusCodeResult(501, "Server Error");
        }

        public ActionResult GetJavaScript() // or JavaScriptResult
        {
            return JavaScript("...JavaScript...");
        }

        public ActionResult GetJson() // or JsonResult
        {
            var obj = new { prop1 = 1, prop2 = "data" };
            return Json(obj, JsonRequestBehavior.AllowGet);
        }

        public ActionResult RedirectTo() // or RedirectResult
        {
            return RedirectPermanent("http://www.site.com");
            //return RedirectToAction("Home", "Index");
        }

        public ActionResult ShowView() // or ViewResult
        {
            return View();
        }
    }
}
```

چند نکته در این مثال وجود دارد:

1) مثلا متد GetJavaScript را درنظر بگیرید. در این متد خاص، چه بنویسید public ActionResult GetJavaScript یا بنویسید public JavaScriptResult GetJavaScript تفاوتی نمی‌کند. در سایر موارد هم به همین ترتیب است. علت را در تصویر سلسله مراتبی ActionResultها می‌توان جستجو کرد. تمام این کلاس‌ها نوعی ActionResult هستند و از یک کلاس پایه به ارث رسیده‌اند.

2) مثلا ContentResult شبیه به همان Response.Write سابق ASP.NET عمل می‌کند. علت وجودی آن هم عدم وابستگی مستقیم به شیء Response و ساده‌تر سازی نوشتن آزمون‌های واحد برای این نوع اکشن متدها است.

3) منهای متد آخری که نمایش داده شده (ShowView)، هیچکدام از متدهای دیگر نیازی به View متناظر ندارند. یعنی نیازی نیست تا روی متد کلیک راست کرده و Add view را انتخاب کنیم. چون در همین متد کنترلر، کار Response به پایان می‌رسد و مرحله بعدی ندارد. مثلا در حالت return File، یک فایل به درون مرورگر کاربر Flush خواهد شد و تمام.

4) متد Welcome و متد Index در اینجا به یک صورت تفسیر می‌شوند. به این معنا که اگر خروجی متد تعریف شده در یک کنترلر از نوع ActionResult نباشد، به صورت پیش فرض درون یک ContentResult محصور خواهد شد.

5) اگر خروجی متدی در اینجا از نوع void باشد، با ActionResult ایی به نام EmptyResult یکسان خواهد بود. بنابراین با متدهای SendMail و SendMailCompleted به یک نحو رفتار می‌گردد.

6) return Json یاد شده که خروجی‌اش از نوع JsonResultاست در پیاده سازی‌های Ajax ایی کاربرد دارد.

7) جهت بازگرداندن حالت وضعیت 403 یا غیرمجاز می‌توان از return new HttpUnauthorizedResult استفاده کرد.

8) یا جهت اعلام مشکلی در سمت سرور به کمک return new HttpStatusCodeResultکد ویژه‌ای را می‌توان به کاربر نمایش داد.

9) به کمک return RedirectToAction می‌توان به یک کنترلر و متدی خاص در آن، کاربر را هدایت کرد.

و خلاصه اینکه تمام کارهایی را که پیشتر در ASP.NET Web forms ، مستقیما به کمک شیء Response انجام می‌دادید (Response.Write، Response.End، Response.Redirect و غیره)، اینبار به کمک یکی از ActionResultهای یاد شده انجام دهید تا بتوان بدون نیاز به راه اندازی یک وب سرور، برای متدهای کنترلرها آزمون واحد نوشت. برای مثال:

```
[TestMethod]
public void TestMethod1()
{
    // Arrange
    var controller = new ActionResultsController();

    // Act
    var result = controller.Index() as ContentResult;

    // Assert
    Assert.NotNull(result);
    Assert.AreEqual( "Hello, World", result.Content);
}
```

نظرات خوانندگان

نویسنده: علی قمشلویی
تاریخ: ۱۳۹۱/۰۱/۱۱ ۱۸:۳۶:۱۹

سلام و تشکر
لطف کنید یکم بیشتر در مورد Response توضیح دهید

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۱/۱۱ ۱۹:۰۹:۱۸

یک سری از اشیاء، اشیاء توکار ASP.NET هستند مانند Application، Server، Response، Request و Session و در فضای نام System.Web تعریف شده‌اند. این اشیاء جزو ASP.Net Runtime هستند و در تمام فریم ورک‌هایی که بر این پایه تهیه شده‌اند قابل دسترسی هستند.
برای نمونه کار شیء Response نمایش اطلاعات به کاربر، تغییرات اعمالی بر روی هدر ارسالی (مثلا ارسال هدر وضعیت 403 یا 404 و امثال آن)، ارسال کوکی‌ها، تنظیم کش و یا انتقال کاربر به مکانی دیگر است (^) .
این نوع اشیاء برای اینکه قابل استفاده باشند نیاز است تا یک وب سرور داشته باشیم، در غیراینصورت نال خواهند بود. برای مثال اگر متد Response.Write در حین یک Unit test فراخوانی شود، قابل آزمایش نخواهد بود مگر اینکه مراحل ست آپ وب سرور و وهله سازی HttpContext طی شود (که کار پر دردسری است). MVC یک لایه abstraction بر روی این اشیاء ایجاد کرده تا در حین انجام آزمون‌های واحد درگیر این مراحل نشویم.

نویسنده: Reza.B
تاریخ: ۱۳۹۱/۰۱/۱۱ ۲۲:۱۸:۱۷

با درود، ممنون بابت نظم و پیگیری و دقت نظر در ارائه مطالب مخصوصا این سری ام‌وی‌سی که امیدوارم سریعتر ادامه بدین.
یک سوال: در قسمت دوم مقاله‌های MVCتون ظاهرا بود که گفتین، هدف این فریم‌ورک MVC معطوف به UI/UX هست. ولاغیر.
لطف میکنید تکنولوژی‌ها یا فریم‌ورک‌های مطرح و لازم برای ASP.NET MVC را که به عنوان مکمل، لازم‌الوجود و حتمی‌الحضور هستند را، معرفی کنید؟ که بصورت موازی دنبال آنها هم باشیم؟؛ در ادامه این سوالم، اخیرا خیلی میبینم کتابخانه‌هایی با پسوند .js ظهور کردن. آیا اینها همان هدف MVC را نشانه گرفته‌اند و یا اگر نه، چقدر مرتبط و کمک‌کننده هستند؟.
با تشکر مجدد.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۱/۱۱ ۲۳:۳۳:۵۶

- به قسمت کار با Ajax در ASP.NET MVC که برسیم، کتابخانه برگزیده، jQuery است. بنابراین لازم است از همین الان اطلاعاتی را در این مورد داشته باشید (^) .
- برای طراحی CSS بین برنامه نویس‌های دات نت، فریم ورکی به نام LESS خیلی محبوبیت دارد (^) .
- کتابخانه‌های جاوا اسکریپتی سمت کلاینت معنا پیدا می‌کنند. بنابراین آنچنان وارد بحث ASP.NET که سمت سرور است نمی‌شوند مگر اینکه کمک حالی در این رابطه باشند مانند jQuery. یا البته جاوا اسکریپت سمت سرور هم به نام نودجی‌اس وجود دارد که بحث دیگری است. در کل حین کار با جاوا اسکریپت دست بازتر است چون داخل مرورگر کاربر اجرا می‌شود که stateful است (مثل برنامه‌های سیلورلایت). به همین جهت کتابخانه MVVM هم برای جاوا اسکریپت وجود دارد (^) . جالب اینجا است که این کتابخانه MVVM توسط یکی از اعضای تیم ASP.NET MVC طراحی شده.

نویسنده: Hooshmand Meysam
تاریخ: ۱۳۹۱/۰۱/۱۲ ۰۰:۵۱:۴۶

با سلام و تشکر از مطالب محشرتان
ممکنه یک سر نخ در مورد UI/UX بدهید؟قبلا هم دیدم، منتهی خوب متوجه نشدم، در حقیقت مطلب جمع و جور و مفیدی ندیدم ازش. با سپاس از لطف شما

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۱/۱۲ ۰۱:۲۵:۳۲

شاید مطالبی که من عنوان کردم این برداشت را به وجود آورده که MVC در مورد UI/UX بحث می‌کند؟
پاسخ اینکه، خیر. بحث ما در اینجا برنامه نویسی وب است نه طراحی رابط کاربری. MVC نظری در مورد دومین شما، در مورد لایه بندی دسترسی به داده‌ها، در مورد استفاده از ORM و غیره ندارد. به همین ترتیب در مورد اینکه ظاهر برنامه رو هم به چه نحوی طراحی کنید، نظری ندارد. اینجا شما آزاد هستید که مطابق روش‌های دیگری که فکر می‌کنید مفید هستند عمل کنید. بحث ما در اینجا جدا سازی منطق برنامه از لایه نمایشی است، برای اینکه بتونیم اون رو تست کنیم. در وب فرم‌ها این منطق به لایه نمایشی گره خورده. در MVC این دو از هم جدا شدن.
بنابراین به صورت خلاصه MVC نظری در مورد نحوه طراحی رابط کاربری و همچنین نحوه دسترسی به اطلاعات ندارد. نمیاد به شما بگه باید از مثلا EF استفاده کنید یا NH . یا اینکه از فلان فریم ورک CSS باید استفاده کنید یا خیر. دسترسی پذیری سایت شما چگونه باید باشد. ترکیب رنگ‌های آن چگونه باید باشد. این‌ها خارج از بحث MVC هستند.
در مورد UI/UX یک سری سایت و وبلاگ فعال و خوب هستند که به نظرم دنبال کردن اون‌ها خیلی مفید است. مثلا:

uxbooth

uxmag

uxmovement

smashingmagazine

نویسنده: Salehi
تاریخ: ۱۳۹۱/۰۱/۱۲ ۰۲:۰۳:۵۷

این بحث رو من جاهای دیگه هم مطرح کردم ولی جواب مناسبی نگرفتم. ببینید توی web form طراح میتونه بیاد UI رو بوسیله کنترل ها طراحی کنه ، بوسیله css به ظاهر اونها برسه و ... بدون اینکه هیچ دانشی از سی شارپ یا Asp.net داشته باشه. بعدش هم برنامه نویس کارش رو انجام بده. البته در این مرحله چون توی کد برنامه نیاز به ارتباط با عناصر صفحه وجود داره وابستگی پیش میاد.
ولی به نظر من توی mvc قضیه برعکسه. وقتی قرار واسط کاربر طراحی بشه ، و از روش ترجیح داده شده strongly type view استفاده بشه (که واقعا جالبه و کار باهاش راحت) ، طراح حتی باید مدل ها رو بشناسه، یا حتی در روش باز هم توصیه شده ، کلاس های سی شارپی به عنوان viewModel تعریف کنه و view رو با اونها تشکیل بده. در حالیکه در مرحله کدنویسی دیگه به عناصر صفحه وابستگی وجود نداره.
منظورم اینه که توی هرکدوم به نظرم وابستگی وجود داره، ولی تو مراحل متفاوت.اینطوره؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۱/۱۲ ۰۸:۳۲:۲۶

کسی که دانشی در مورد وب فرم‌ها نداره چطور می‌تونه مثلا ستون‌های یک GridView رو طراحی و بایند کنه؟ برای نمونه یک گزارش رو دربیاره؟ اینجا هم به همین ترتیب. یک ترکیب کلی صفحه وجود دارد که طراح می‌تونه صرفنظر از اینکه کار شما PHP است یا ASP.NET در بیاره. مثلا یک فایل PSD به شما تحویل بده (روش مرسوم). یک قسمت‌هایی هم باقی خواهد ماند که باید برنامه نویس پرکنه زمانیکه این فایل PSD رو تبدیل به قالب سایت کرد. زمانیکه master page رو درست کرد. المان‌ها رو در جاهای مختلف جایگذاری کرد.

نویسنده: Salehi
تاریخ: ۱۳۹۱/۰۱/۱۲ ۱۰:۱۳:۵۴

مثال gridview جالب بود. ولی خب ،پس به هرحال درسته که توی mvc هم view به طور کامل از بقیه قسمت ها جدا نمیشه.
البته دیدگاه دیگه ای هم وجود داره: این جداسازی تا جایی مهم و موردنظره که نوشتن آزمون های واحد رو راحت تر کنه.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۱/۱۲ ۱۰:۱۹:۵۷

ببینید این یک «سیستم» هست. سیستم هم مجموعه‌ای است از عناصر (اجزاء) که برای رسیدن به یک هدف واحد و مشخص با هم همکاری می‌کنند. بنابراین اگر عنوان شود این View خارج و مستقل از این سیستم معنا پیدا می‌کند، کمی زیاده روی است.
ولی این جدا سازی یک منفعت رو به همراه داره. چون کنترلر ارجاع مستقیمی به اشیاء بصری نداره، برنامه نویس می‌تونه کارش رو بدون نیاز به View پیش ببره و نهایتا یکپارچه کنه.

نویسنده: Ahmad
تاریخ: ۱۳۹۱/۰۱/۱۶ ۱۸:۰۵:۴۳

فکر کنم منظورتون این فریم ورک بود lessframework.com

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۱/۱۶ ۱۹:۰۴:۲۲

منظورم بیشتر این بود البته: http://www.dotlesscss.org/

نویسنده: محمد شهریاری
تاریخ: ۱۳۹۱/۰۴/۰۴ ۱۲:۵

سلام
معنی جمله شما که تمامی Action ها به جز(ShowView) نیاز به View متناظر ندارند و در همین متد کنترلر کار Response به پایان میرسد را به درستی متوجه نشدم . البته در بخش قبل از RedirectToAction استفاده کردید . می‌خواستم بدونم منظور شما نیز همین می‌باشد و اگر به این صورت است فرق این Action ها با متد عادی چیست ؟

با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۰۴ ۱۲:۲۰

زمانیکه return View داشته باشید، MVC به دنبال View هم نام با متد جاری خواهد گشت. البته می‌شود در اینجا viewName را هم دستی تعیین کرد ولی اگر تعیین نشود از نام متد استفاده می‌شود.
در سایر حالت‌های یاد شده برای نمونه زمانیکه return File فراخوانی می‌شود یا موارد مشابه، در پشت صحنه در آخر کار متد Response.End فراخوانی خواهد شد. یعنی مثلا یک خروجی مشخص به درون مرورگر کاربر Flush شده و درخواست خاتمه می‌یابد. بنابراین در اینجا نیازی به View متناظر با متد نیست چون کار تمام شده است.

نویسنده: وحید مختاری
تاریخ: ۱۳۹۱/۰۵/۲۰ ۱۵:۱۵

مراحل ست آپ سرورمنظور چیست ؟
یعنی MVC پیشفرض نیازی به وب سرور ندارد باتشکر.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۲۰ ۱۶:۱۷

اگر به TestMethod ایی که در انتهای بحث مطرح شده دقت کنید، بدون نیاز به برپایی یک وب سرور، قابل آزمایش است. یعنی آزمودن ContentResult نیازی به طی شدن مراحل تشکیل HttpContext ندارد.

نویسنده: ناشناس
تاریخ: ۱۳۹۲/۰۸/۰۳ ۲۱:۵۶

سلام -
برای آرگومان دوم ("خطا ... "HttpStatusCodeResult("501","پیام فارسی میذارم بجاش ؟ نشون میده- یونیکد رو پشتیبانی نمی‌کنه یا کار خاصی باید انجام بدم؟
مرسی

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۸/۰۴ ۰:۴۴

مطابق استاندارد ، در HTTP header امکان قرار دادن کاراکترهای یونیکد نیست (پیش فرض آن حداکثر ISO-8859-1 است یا حروف لاتین):

```
Reason-Phrase  = *<TEXT, excluding CR, LF>
“The TEXT rule is only used for descriptive field contents and values that are not intended to be
interpreted by the message
parser. Words of *TEXT MAY contain characters from character sets other than ISO-8859-1
only when encoded according to the rules of RFC 2047”.
```

مگر اینکه مطابق RFC 2047 انکد شوند. (از این RFC هم بیشتر در عنوان ایمیل‌ها تابحال استفاده شده تا در هدر HTTP)
البته می‌شود توسط HttpUtility.UrlEncode این پیام را encode و در سمت کلاینت توسط مثلا jQuery با استفاده از متد استاندارد decodeURIComponent آن‌را دریافت کرد ولی ... به صورت پیش فرض و encode نشده، تفسیر نمی‌شود و حتی به عنوان یک هدر مخرب شاید برگشت زده شود.

نویسنده: پوریا
تاریخ: ۱۳۹۳/۰۷/۲۰ ۱۳:۱۴

سلام و تشکر؛ میشه لطفا توضیح بدید چرا توی بعضی از بازگشت‌ها مثل EmptyResult و HttpUnauthorizedResult وهله سازی انجام شده (new) و در باقی نیازی به این امر نیست.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۷/۲۰ ۱۳:۲۴

در اصل تمام این‌ها نیاز به وهله سازی دارند. اما برای ساده‌تر شدن کار، در کلاس پایه Controller، این وهله سازی‌ها صورت گرفته (^) و یک متد اضافی برای آن‌ها تعریف شده‌است. به این صورت به نظر می‌رسد که وهله سازی انجام نمی‌شود؛ ولی در اصل این‌کار محول شده به متدهای کمکی کلاس پایه کنترلر. برای مثال:

```
protected internal virtual JavaScriptResult JavaScript(string script)
{
    return new JavaScriptResult { Script = script };
}
```

در اینجا return new، داخل یک متد کلاس پایه محصور شده‌است.