عنوان: **ASP.NET Web API - قسمت اول**
نویسنده: بهروز راد
تاریخ: ۱۳۹۱/۰۴/۱۱ ۸:۴
آدرس: www.dotnettips.info
گروه‌ها: WCF, ASP.NET Web API, ASMX, Web Service

بخش هایی از کتاب "مرجع کامل ASP.NET MVC (با پوشش کامل ASP.NET MVC 4)"
ترجمه و تالیف: بهروز راد
وضعیت: در دست چاپ

Web API چیست؟

Web API، نوع قالب جدیدی برای پروژه‌های مبتنی بر وب در .NET. است که بر مبنای اصول و الگوهای موجود در ASP.NET MVC ساخته شده است و همراه با ASP.NET MVC 4 وجود دارد. Web API توسعه گران را قادر می‌سازد تا با استفاده از یک الگوی ساده که در Controllerها پیاده سازی می‌شود، وب سرویس‌های مبتنی بر پروتوکل HTTP را با کدها و تنظیمات کم ایجاد کنند. این سبک جدید برای ایجاد وب سرویس ها، می‌تواند در انواع پروژه‌های .NET. مانند ASP.NET MVC، ASP.NET Web Forms، Windows Application و ... استفاده شود.

یک سوال کاملاً منطقی در اینجا به وجود می‌آید. چرا نیاز به بستری جدید برای ایجاد وب سرویس داریم؟ آیا در حال حاضر مایکروسافت بستری محبوب و فراگیر برای توسعه‌ی وب سرویس هایی که بتوانند با پروتوکل SOAP تعامل داشته باشند در اختیار ندارد؟ مگر وب سرویس‌های ASMX از زمان معرفی ASP.NET وجود نداشته اند؟ آیا تکنولوژی WCF مایکروسافت، بیشترین انعطاف پذیری و قدرت را برای تولید وب سرویس‌ها در اختیار قرار نمی‌دهد؟ وب سرویس‌ها جایگاه خود را یافته اند و توسعه گران با تکنولوژی‌های موجود به خوبی آنها را پیاده سازی و درک می‌کنند. چرا Web API؟

چرا Web Api؟

برای پاسخ به این سوال، باید برخی مشکلات را بررسی کنیم و ببینیم ابزارهای موجود چه راه حلی برای آنها در نظر گرفته اند. اگر با گزینه هایی که در ادامه می‌آیند موافق هستید، خواندن این مطلب را ادامه دهید، و اگر اعتقادی به آنها ندارید، پس نیازهای شما به خوبی با بسترهای موجود پاسخ داده می‌شوند.

من معتقد هستم که راه بهتری برای ایجاد وب سرویس‌ها وجود دارد.

من معتقد هستم که روش‌های ساده‌تری برای ایجاد وب سرویس‌ها وجود دارد و WCF بیش از حد پیچیده است.

من معتقد هستم که تکنولوژی‌های پایه‌ی وب مانند افعال GET، POST، PUT و DELETE برای انجام اعمال مختلف توسط وب سرویس‌ها کافی هستند.

اگر همچنان در حال خواندن این مطلب هستید، توضیحات خود را با شرح تفاوت میان Web API و تکنولوژی‌های دیگر هم حوزه‌ی آن ادامه می‌دهیم و خواهید دید که استفاده از Web API چقدر آسان است.

تفاوت Web API و WCF

وب سرویس‌های ASMX تا چندین سال، انتخاب اول برای ایجاد وب سرویس‌های مبتنی بر پروتوکل SOAP با استفاده از پروتوکل HTTP بودند. وب سرویس‌های ASMX، از وب سرویس‌های ساده که نیاز به قابلیت تعامل پایین داشتند و در نتیجه به پروتوکل SOAP نیز وابسته نبودند پشتیبانی نمی‌کردند. WCF جای وب سرویس‌های ASMX را گرفت و خود را به عنوان آخرین و بهترین روش برای ایجاد وب سرویس‌ها در بستر .NET. معرفی کرد. نمونه ای از یک سرویس WCF بر مبنای پروتوکل HTTP در .NET. به صورت ذیل است.

```
[ServiceContract]
public interface IService1
{
    [OperationContract]
    string GetData(int value);
    [OperationContract]
    CompositeType GetDataUsingDataContract(CompositeType composite);
}
...
public class Service1 : IService1
{
    public string GetData(int value)
    {
        return string.Format("You entered: {0}", value);
```

```
    }

    public CompositeType GetDataUsingDataContract(CompositeType composite)
    {
        if (composite == null)
        {
            throw new ArgumentNullException("composite");
        }
        if (composite.BoolValue)
        {
            composite.StringValue += "Suffix";
        }
        return composite;
    }
}
```

در WCF، پایه و اساس وب سرویس را یک interface تشکیل می‌دهد. در حقیقت اجزای وب سرویس را باید در یک interface تعریف کرد. هر یک از متدهای وب سرویس در interface تعریف شده که صفت OperationContract برای آنها در نظر گرفته شده باشد، به عنوان یکی از اعمال و متدهای قابل فراخوانی توسط استفاده کننده از وب سرویس در دسترس هستند. سپس کلاسی باید ایجاد کرد که interface ایجاد شده را پیاده سازی می‌کند. در قسمت بعد، با مفاهیم پایه‌ی Web API و برخی کاربردهای آن در محیط ASP.NET MVC آشنا می‌شوید.

نتیجه گیری

Web API، یک روش جدید و آسان برای ایجاد وب سرویس ها، بر مبنای مفاهیم آشنای ASP.NET MVC و پایه‌ی وب است. از این روش می‌توان در انواع پروژه‌های .NET. استفاده کرد.

نظرات خوانندگان

نویسنده: یوسف نژاد
تاریخ: ۱۳۹۱/۰۴/۱۱ ۱۱:۵۹

این مقوله خیلی مفیده و کاربردی هست. خیلی وقت بود میخواستم در موردش بیشتر تحقیق کنم. با تشکر بابت زحماتتون و آغاز این سری مطلب جدید.

نویسنده: شهروز جعفری
تاریخ: ۱۳۹۱/۰۴/۱۱ ۱۶:۷

زمان انتشار کی هس؟

نویسنده: بهروز راد
تاریخ: ۱۳۹۱/۰۴/۱۱ ۱۹:۵

بستگی به ناشر داره. اما نباید بیشتر از دو هفته طول بکشه.

نویسنده: شهروز جعفری
تاریخ: ۱۳۹۱/۰۴/۱۱ ۱۹:۸

نظر کلی من اینه که نمیشه به wcf گفت پیچیده آخه هدفش فرق میکنه.و در ضمن مگه APIحدوداً همون Rest با معماری خیلی ساده‌تر نیست؟

نویسنده: شهروز جعفری
تاریخ: ۱۳۹۱/۰۴/۱۱ ۱۹:۸

بیصبرانه منتظرش هستم

نویسنده: بهروز راد
تاریخ: ۱۳۹۱/۰۴/۱۱ ۲۲:۴۷

REST بیشتر برای مواقعی هست که شما عملیات CRUD انجام میدید. در حالی که با Web API می‌تونید علاوه بر CRUD، کارهای بسیار بیشتری انجام بدید. مفاهیم و قابلیت‌های موجود در ASP.NET MVC مانند فیلترها به خوبی در Web API پشتیبانی و به راحتی قابل استفاده هستند. ضمن اینکه با Web API می‌تونید معماری REST رو با تغییر کوچکی در route پیش فرض به دست بیارید و بدین شکل، مهاجرت از REST به Web API بسیار راحت هست. در اوایل معرفی Web API، از پروتوکل OData نیز پشتیبانی اولیه میشد که متاسفانه مایکروسافت در نسخه‌ی RC این پشتیبانی رو حذف کرد. شاید در نسخه‌های بعدی این قابلیت نیز اضافه بشه که به قدرتمندتر شدن Web API کمک می‌کنه.
ضمناً، پشتیبانی مایکروسافت از WCF REST API نیز به اتمام رسیده و پیشنهاد شده که از Web API استفاده کنید.
http://aspnet.codeplex.com/wikipage?title=WCF%20REST

نویسنده: رضا.ب
تاریخ: ۱۳۹۱/۰۴/۱۲ ۳:۳

در بند سوم اشاره کردین : من معتقد هستم که تکنولوژی‌های پایه‌ی وب مانند أفعال GET، POST، PUT و DELETE برای انجام أعمال مختلف توسط وب سرویس‌ها کافی هستند. ا گر ضروری نیستند بیشتر از CRUD باشند پس خاصیت ویژه‌ای که شما میگین "کارهای بسیار بیشتری" میتونه انجام بده چی هست که WCF پاسخگو نیست؟

در ضمن فکر میکنم REST فقط با منابع و ورب‌های HTTP کار داره. و برای همین سهولت و سادگیش پروتکول SOAP نسخه منسوخ

شده‌ی وب‌سرورها به‌حساب میاد. اینطور نیست؟
سوال دیگه‌ام در مورد میزان نقش Web API هست. آیا رسالت واقعی یک وب‌سرویس رو هدف گرفته؟ یعنی پیاده سازی یک Endpoint که شامل یه‌سری interface هستند که امکاناتی رو در اختیار کلاینت قرار میده؟
ممنون از توجه‌تون.

نویسنده: بهروز راد
تاریخ: ۱۳۹۱/۰۴/۱۲ ۸:۳۷

دقت داشته باشید که Web API عرضه نشده تا WCF رو منسوخ کنه. برنامه هایی که صرفاً از بستر پروتوکل HTTP به عنوان یک سرویس برای رد و بدل کردن داده‌ها استفاده می‌کنند، بهتره که از این به بعد از Web API استفاده کنند. ضمن سادگی و مفاهیم آشنای ASP.NET MVC، روش یکپارچه ای برای ایجاد وب سرویس‌های HTTP نیز به وجود اومده که مشکلات استفاده از WCF رو از بین می‌بره. WCF ذاتاً برای پیغام‌های SOAP محور طراحی شده و به کار گرفتن اون برای وب سرویس‌های HTTP یا به زور خوراندن HTTP به اون بی معنیه. در WCF راه‌های مختلفی برای ایجاد وب سرویس‌های HTTP وجود داره که باعث گمراهی و سردرگمی توسعه گر میشه و حتی فریمورک‌های مختلفی مانند OpenRasta و ServiceStack نیز بدین منظور وجود دارند. بنابراین پشتیبانی WCF از HTTP به یک پروژه‌ی دیگه تحت نام ASP.NET Web API منتقل شده و WCF Web API دیگه پشتیبانی نمیشه. کمی تغییر نام و کمی جابجایی مفاهیم دراینجا صورت گرفته. WCF همچنان قدرتمنده و نباید Web API به هیچ وجه به عنوان جایگزینی برای اون تصور بشه. ایجاد بسترهایی برای ارتباطات دو طرفه یا صفی از پیغام‌ها یا سویچ بین کانال‌ها در هنگام فعال نبودن یک کانال، اینها همه از قابلیت‌هایی هست که Web API هرگز جایگزینی برای اونها نخواهد بود و مختص WCF هستند.

نویسنده: ramin
تاریخ: ۱۳۹۱/۰۴/۱۲ ۹:۲۶

سلام آقای راد
ببخشید که سوال بی ربط رو اینجا میپرسم
آیا برنامه ای برای انتشار ویرایش جدید کتاب Entity Framework دارین؟

نویسنده: بهروز راد
تاریخ: ۱۳۹۱/۰۴/۱۲ ۱۰:۱۴

لطفاً سوالات اینچنینی رو از طریق ایمیل behrouz.rad[atsign]gmail بپرسید.
بله، بعد از کتاب ASP.NET MVC، کتاب Entity Framework رو آپدیت می‌کنم.

نویسنده: torisoft
تاریخ: ۱۳۹۱/۰۴/۱۶ ۲۳:۵

سلام جناب راد
از Web API تو سیلورلایت هم میشه استفاده کرد ؟
اگه استفاده میشه آیا مثبت میدونید استفاده از اونو تو سیلور ؟
با تشکر

نویسنده: بهروز راد
تاریخ: ۱۳۹۱/۰۴/۱۷ ۸:۱۱

بله مشکلی نداره. پروژه‌ی Silverlight رو در یک پروژه‌ی وب Host کنید.
Silverlight هم یک نوع پروژه است، مثل Web و Desktop. اگر پروژه‌ی شما بر مبنای Silverlight هست و نیاز دارید تا امکانات اون رو به صورت سرویس ارائه بدید، می‌تونید از Web API برای عرضه‌ی این امکانات استفاده کنید.

نویسنده: حمید
تاریخ: ۱۳۹۱/۰۴/۱۷ ۱۱:۲۲

سلام. وقت بخیر.
مطالب خیلی خوب و به روزی دارین و خدا قوت..
با عرض معذرت می‌خواستم بگم من MVC4 رو نصب کردم اما بازم بعد انتخاب MVC4 از لیست Templateهای ویژوال استودیو گزینه Web API رو مشاهده نمی‌کنم.آیا افزونه یا برنامه خاصی باید نصب کنم.از قبل از زحمتتون تشکر می‌کنم.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۱۷ ۱۱:۵۰

مراجعه کنید به قسمت دوم ، تصویر سوم

نویسنده: حمید
تاریخ: ۱۳۹۱/۰۴/۲۰ ۲۲:۵۷

سلام.مشکل من همینه که همین تصویر سوم رو که میگین تو این بخش من گزینه Web API رو ندارم.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۲۰ ۲۳:۱۱

از طریق NuGet هم می‌تونید برای نصب آن اقدام کنید. این رو هم تست کنید:
http://nuget.org/packages/aspnetwebapi

نویسنده: Saeed M. Farid
تاریخ: ۱۳۹۱/۰۴/۲۱ ۱۱:۳

سلام و ممنون از مطلب مفید:

امکانش هست در مورد *"سویچ بین کانال‌ها در هنگام فعال نبودن یک کانال"* کمی بیشتر راهنمایی کنید یا مرجع (لینک) معرفی کنید؟ من از صحبت شما اینطور برداشت کردم که میشه در channel shape (های)ی که مثلاً برای duplex communications (یعنی IDuplexChannel) پیاده سازی کردم، اگه چنین کانالی در دسترس نبود، سوئیچ کنه روی طراحی مبتنی بر one-way messaging من؟ اصلاً چنین امکانی در سطح IChannelListener هست یا ChannelFactory ؟ کلاً اگه ممکنه یه توضیح کلی در مورد چنین امکانی که در موردش صحبت کردین بدین یا اگه جایی در موردش قبلاً بحث شده (که حتماً شده!) من رو هدایت کنید به اون، چون گلوگاه سیستم‌هام همین مورد هست.
پیشاپیش ازتون ممنونم...

نویسنده: بهروز راد
تاریخ: ۱۳۹۱/۰۴/۲۱ ۱۹:۲۹

اصولاً در Web API چیزی با عنوان Channel با اون مفهوم که در WCF هست نداریم. در Web API فقط یک Transport Channel برای HTTP وجود داره، چون هدف ایجاد Web API، فقط برقراری ارتباط در سطح HTTP هست، نه مثلاً MSMQ.
Protocol Channel هم همان مفاهیمی هستند که در ASP.NET MVC وجود دارند و مثلاً قسمتی از اون، تصدیق هویت و تعیین مجوز کاربر برای دسترسی به منابع با استفاده از فیلتر Authorize هست.
لطفاً دنبال تطبیق و تناظر بین مفاهیم پیچیده‌ی WCF و یافتن معادل در Web API نباشید. Web API به وجود آمده تا ایجاد وب سرویس‌ها در بستر HTTP رو ساده کنه، همین!

نویسنده: سیروس
تاریخ: ۱۳۹۱/۱۲/۰۲ ۱۸:۳۳

سلام
یک سوال مهم داشتم، آیا استفاده از web api در Windows Form مانند WCF ممکن است، یعنی پروژه ما هم هاست و هم کلاینت رو MVC یا ASP.Net نیست، اگه میشه یه منبع معرفی کنید.

نویسنده: محسن
تاریخ: ۱۳۹۱/۱۲/۰۲ ۲۲:۳۶

اولین نتیجه جستجوی گوگل در مورد winforms web api :

Using Microsoft Web API from a Windows and WinRT Client Application

نویسنده: محمد آزاد
تاریخ: ۱۳۹۱/۱۲/۰۲ ۲۲:۳۸

تو مقدمه به این مطلب اشاره شده دوست عزیز
این سبک جدید برای ایجاد وب سرویس ها، می‌تواند در انواع پروژه‌های .NET مانند ASP.NET MVC، ASP.NET Web Forms، Windows Application و ... استفاده شود.

نویسنده: سیروس
تاریخ: ۱۳۹۱/۱۲/۰۵ ۱۰:۱۰

محسن => دوست عزیز من اون مطلب رو قبلا هم مطالعه کردم، قسمت هاست رو MVCست.اینقدر بی سواد نیستم که نتونم سرچ کنم.
آزاد => میدونم که تو Win APP قابل استفاده هست، اما می‌خوام بدونم پروژه هاست مثل WCF می‌تونه رو مستقل از Asp.Net باشه یا نه چون ظاهرا پیاده سازی WebAPI فقط روی ASP.Net امکان پذیر است.

نویسنده: محسن
تاریخ: ۱۳۹۱/۱۲/۰۵ ۱۰:۱۶

نتیجه جستجوی گوگل در مورد wep api self host :

Self-Host a Web API

نویسنده: یاسر مرادی
تاریخ: ۱۳۹۱/۱۲/۰۵ ۱۲:۳۶

من فکر کنم مطلب این دوستمون رو این جوری مطرح کنم بهتره
وقتی شما از WCF Data Services استفاده می‌کنید، WCF Data Services Client دارید، که به شما امکان نوشتن کوئری‌های Linq در سمت کلاینت، Change Tracking و Merge و ... رو می‌ده
اما من همچین آیتمی رو برای Web API پیدا نکردم، بهترین چیزی که دیدم Http Client بوده که در حد مثال زدن خوبه، ولی به درد پروژه نویسی نمی‌خوره، این که شما یک کلاینت قوی داشته باشید، خیلی مهمه، Http Client تفاوت مفهومی زیادی با $.ajax نداره
حتی در JayData هم همین طور هستش، و شما پشتیبانی خیلی بهتری از WCF Data Services می‌بینید تا از Web API، همین طور در Breeze.js
در اندروید و iOS هم شما پشتیبانی WCF Dat Services Client رو دارید، ولی Web API خیر
موفق باشید

نویسنده: محسن
تاریخ: ۱۳۹۱/۱۲/۰۵ ۱۲:۵۸

سؤال مطرح شده در مورد هاست کردن یک سرویس در برنامه ویندوزی بود که اصطلاحا Self hosting نام دارد.

Web API امکان استفاده از OData را هم دارد:

Getting started with ASP.NET Web API OData in 3 simple steps

نویسنده: یاسر مرادی
تاریخ: ۱۳۹۱/۱۲/۰۵ ۱۳:۳۵

قبول، ولی در هر حال آیا راهی جز Http Client برای دسترسی به Web API وجود دارد ؟

مثلا مبدل Linq به OData ؟
به همراه Change Tracking و ...
در ضمن موارد مهمی از OData مانند $batch در Web API پشتیبانی نشده اند، و باید برایشان Message Formatter نوشت، این نیز کار را سخت می‌کند
بر خلاف نظر دوستان به نظر من به هیچ وجه هیچ فریم ورکی راحت‌تر از WCF Data Services وجود ندارد، که جمعا با 3 خط کد راه اندازی می‌شود.

نویسنده: محسن
تاریخ: ۱۳۹۱/۱۲/۰۵ ۱۷:۲۴

NuGet مربوط به Web API OData مرتبا به روز میشه. آخرین به روز رسانی آن مربوط به 5 روز قبل بوده.

ضمن اینکه خروجی OData استاندارد است. بنابراین با کلاینت‌های موجود کار می‌کنه. فرقی نمی‌کنه تولید کننده چی هست تا زمانیکه استاندارد رعایت بشه.

نویسنده: یاسر مرادی
تاریخ: ۱۳۹۱/۱۲/۰۵ ۱۸:۵

دوست عزیز، فکر کنم سوال من خیلی واضح باشه
مسئله اول این هستش که مواردی از OData هست که در WCF Data Services وجود داره، ولی در Web API خیر، OData یک سری استاندارد هستش، بالاخره باید یک جایی پیاده سازی بشه، مثل HTML 5، که قسمت‌های مختلفش در درصدهای متفاوت در مرورگرهای متفاوت پیاده سازی شده، در این میان Chrome بهتر از IE هستش، چرا ؟ چون استانداردهای بیشتری رو پیاده سازی کرده
دوم این که آیا شما به صورت عملی از Breeze js و Jay Data و WCF Data Services Client استفاده کرده اید ؟ درسته که اینها به OData وصل می‌شوند، ولی میزان امکانات اینها برای WCF Data Services قابل قیاس با Web API نیست.
سوال اصلی من با این تفاسیر این است :
اگر قبول کنیم که راهی برای دسترسی به Web API وجود ندارد، الا استفاده از jQuery Ajax و Http Client، شما به چه صورت یک پروژه بزرگ رو با Web API می‌نویسید ؟
Change Tracking رو چه جوری پیاده سازی می‌کنید ؟
به چه صورت در کلاینت هایی مانند اندروید، و یا Win RT و ... از Linq برای دسترسی به سرویس هاتون استفاده می‌کنید ؟
اگر فرض کنیم که می‌خواهیم یک سرویس عمومی بنویسیم که همه جا به سادگی قابل استفاده باشه، آیا از Web API استفاده می‌کنید ؟
خلاصه : مزیت واقعی Web API چیست و چه زمانی پروژه ای رو با Web API شروع می‌کنید ؟
موفق و پایدار باشید

نویسنده: محسن
تاریخ: ۱۳۹۱/۱۲/۰۵ ۱۸:۳۱

«مزیت واقعی Web API چیست و چه زمانی پروژه ای رو با Web API شروع می‌کنید ؟»

WCF or ASP.NET Web APIs

به علاوه هدف اصلی Web API و یکپارچگی آن با خصوصا MVC (و بعد وب فرم‌ها) در درجه اول توسعه ActionResultهای پیش فرض MVC است (به همین جهت اول اسم آن ASP.NET است و نه مثلا اندروید):

ASP.NET Web API vs. ASP.NET MVC APIs

نویسنده: یاسر مرادی
تاریخ: ۱۳۹۱/۱۲/۰۵ ۱۹:۳

مقاله اول Web API رو با **WCF خام** مقایسه کرده، نه با WCF Data Services

مقاله دوم هم Actionهای Web API رو با MVC قیاس کرده
اگر شما یک مقاله بنویسید که مثلا Web API رو با ASP.NET Web Service قیاس بکنه، و نشون بده مزیت‌های Web API بیشتره، این می‌شه مزیت Web API بر ASP.NET Web Service، نه بر WCF Data Services
ممکنه این موارد هم مهم باشند، ولی اون چیزی که برای من سوال شده این هستش که چه زمانی در یک پروژه WCF Data Services رو می‌گذاریم کنار و از Web API استفاده می‌کنیم ؟
در واقع با توجه به امکانات واقعا زیاد WCF Data Services چرا باید اساسا از Web API استفاده بشه، اگر شما می‌فرمایید که 5 روز پیش برای Web API نسخه آمده، این عدد برای Data Services چهار روز پیش بوده
اگر بحث امکانات هست، لیست زیادی از امکانات رو من شمردم و می‌شه شمرد، از امکاناتی که تو Data Services هست، ولی تو Web API نیست.
اگر من اندروید رو مثال زدم، برای سمت کلاینت بود، شما در اندروید با چی به Web API وصل می‌شید ؟
با jQuery Ajax ؟
یا می‌خواهید به App Serverهای .NET. ای برنامه‌های دیگر، بگویید با Http Client از سرویس‌های شما استفاده کنند ؟
با سپاس

نویسنده: محسن
تاریخ: ۱۳۹۱/۱۲/۰۵ ۱۹:۱۶

هدف مایکروسافت از یکپارچه کردن WEB API با ASP.NET و خصوصا MVC ارائه یک سری ActionResult *Super* است بجای ActionResultهای معمولی MVC3. برای نمونه:

Using Kendo UI grid with Web API and OData

نویسنده: میثم99
تاریخ: ۱۳۹۳/۰۱/۲۱ ۲۳:۰

سلام
می‌خواهم بدانم برای امنیت web api در پروژه های web form چه کارهایی باید انجام دهیم بیشتر مطالب در مورد mvc هست مثلا Anti-Forgery Tokens برای mvc به راحتی می‌توان استفاده کرد ولی برای web form چکار بهتر است انجام دهیم؟
در اینجا ما مستقیما با دستورات post put و delete کار داریم که اطلاعات بانک اطلاعاتی رو تغییر می‌دهند. حالا چطور می‌توان امنیت رو کاملا تامین کرد ؟

مثلا کاربران شناسایی شده اطلاعات را وارد کنند و اینکه شخصی نتواند با یک دستور ای جکس توسط مرورگر اطلاعات اشتباه در سایت ثبت کند؟ و یا هر مشکل امنیتی دیگری که پیش بیاید؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۱/۲۱ ۲۳:۲۱

روش‌های زیادی برای تامین امنیت در وب API و کار با «کاربران شناسایی شده » وجود دارند. لیست رسمی
از این لیست رسمی، دو مورد معروف آن در سایت جاری بررسی شده:

ASP.NET Identity

Forms authentication

مباحث پایه‌ای این‌ها مشترک است بین MVC و وب فرم‌ها و سایر فناوری‌های مشابه.

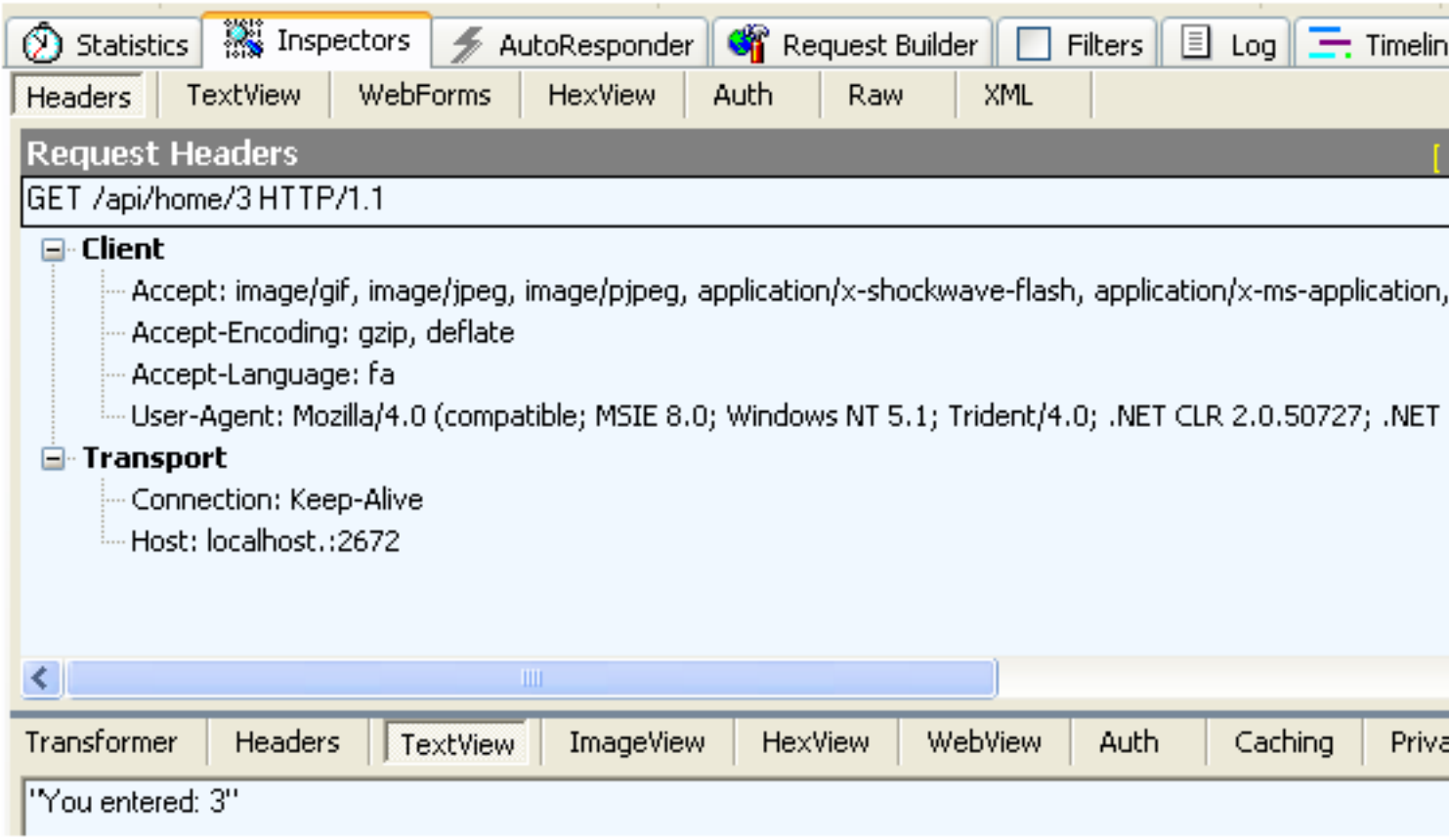
عنوان: **ASP.NET Web API - قسمت دوم**
نویسنده: بهروز راد
تاریخ: ۱۳۹۱/۰۴/۱۳ ۷:۵۸
آدرس: www.dotnettips.info
برچسب‌ها: WCF, ASP.NET Web API, ASMX, Web Service

در قسمت اول به دلایل ایجاد ASP.NET Web API پرداخته شد. در این قسمت، یک مثال ساده از Web API را بررسی می‌کنیم.
تلاش‌های بسیاری توسط توسعه گران صورت پذیرفته است تا فرایند ایجاد وب سرویس WCF در بستر HTTP آسان شود. امروزه وب سرویس هایی که از قالب REST استفاده می‌کنند مطرح هستند.
ASP.NET Web API از مفاهیم موجود در ASP.NET MVC مانند Controllerها استفاده می‌کند و بر مبنای آنها ساخته شده است. بدین شکل، توسعه گر می‌تواند با دانش موجود خود به سادگی وب سرویس‌های مورد نظر را ایجاد کند. Web API، پروتوکل SOAP را به کتاب‌های تاریخی! سپرده است تا از آن به عنوان روشی برای تعامل بین سیستم‌ها یاد شود. امروزه به دلیل فراگیری پروتوکل HTTP، بیشتر محیط‌های برنامه نویسی و سیستم ها، از مبانی اولیه‌ی پروتوکل HTTP مانند افعال آن پشتیبانی می‌کنند.
حال قصد داریم تا وب سرویسی را که در قسمت اول با WCF ایجاد کردیم، این بار با استفاده از Web API ایجاد کنیم. به تفاوت این دو دقت کنید.

```
using System.Web.Http;

namespace MvcApplication1.Controllers
{
    public class ValuesController : ApiController
    {
        // GET api/values/5
        public string Get(int id)
        {
            return string.Format("You entered: {0}", id);
        }
    }
}
```

اولین تفاوتی که مشهود است، تعداد خطوط کمتر مورد نیاز برای ایجاد وب سرویس با استفاده از Web API است، چون نیاز به interface و کلاس پیاده ساز آن وجود ندارد. در Web API، Controllerهایی که در نقش وب سرویس هستند از کلاس ApiController ارث می‌برند. اعمال مورد نظر در قالب متدها در Controller تعریف می‌شوند. در مثال قبل، متد Get، یکی از اعمال است.
نحوه‌ی برگشت یک مقدار از متدها در Web API، مانند WCF است. می‌توانید خروجی متد Get را با اجرای پروژه‌ی قبل در Visual Studio و تست آن با یک مرورگر ملاحظه کنید. دقت داشته باشید که یکی از اصولی که Web API به آن معتقد است این است که وب سرویس‌ها می‌توانند ساده باشند. در Web API، تست و دیباگ وب سرویس‌ها بسیار راحت است. با مرورگر Internet Explorer به آدرس http://localhost:{port}/api/values/3 بروید. پیش از آن، برنامه‌ی Fiddler را اجرا کنید. شکل ذیل، نتیجه را نشان می‌دهد.



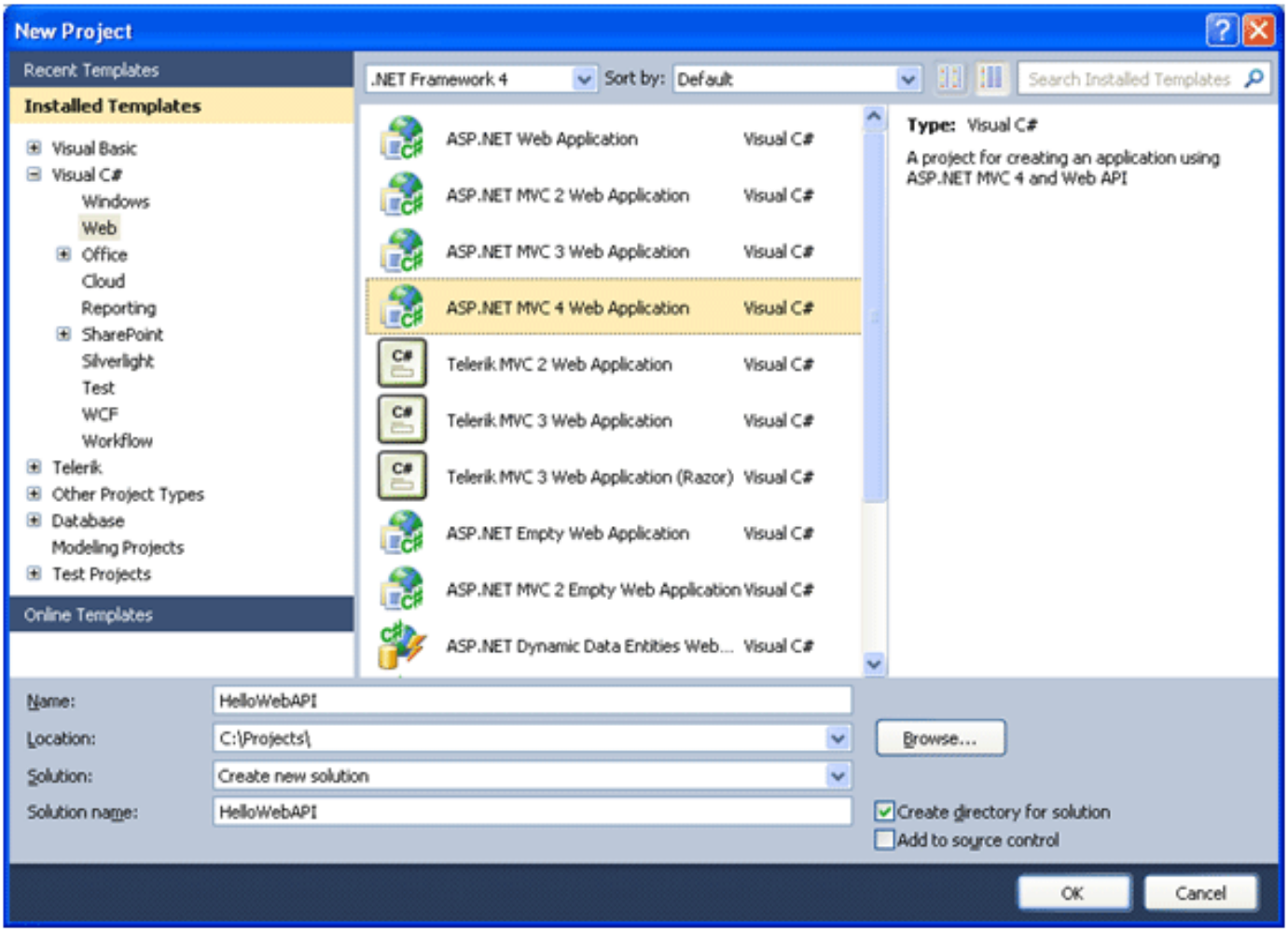
در اینجا نتیجه، عبارت "You entered: 3" است که به صورت یک متن ساده برگشت داده شده است.

ایجاد یک پروژه‌ی Web API

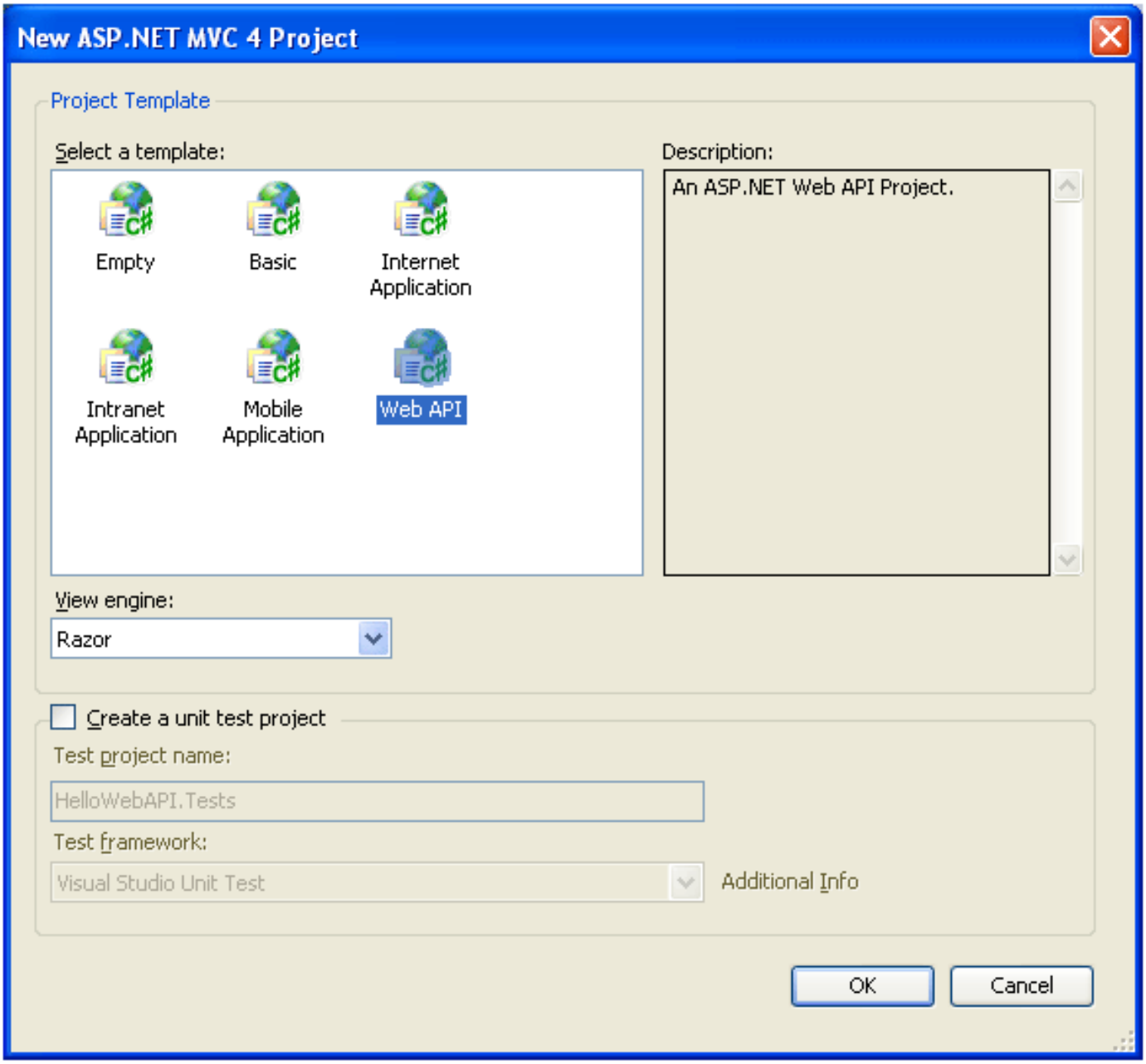
در Visual Studio، مسیر ذیل را طی کنید.

File> New> Project> Installed Templates> Visual C#> Web> ASP.NET MVC 4 Web Application

نام پروژه را HelloWebAPI بگذارید و بر روی دکمه‌ی OK کلیک کنید (شکل ذیل)



در فرمی که باز می‌شود، گزینه‌ی Web API را انتخاب و بر روی دکمه‌ی OK کلیک کنید (شکل ذیل). البته دقت داشته باشید که ما همیشه مجبور به استفاده از قالب Web API برای ایجاد پروژه‌های خود نیستیم. می‌توان در هر نوع پروژه ای از Web API استفاده کرد.



اضافه کردن مدل

مدل، شی ای است که نمایانگر داده‌ها در برنامه است. Web API می‌تواند به طور خودکار، مدل را به فرمت XML، JSON یا فرمت دلخواهی که خود می‌توانید برای آن ایجاد کنید تبدیل و سپس داده‌های تبدیل شده را در بدنه‌ی پاسخ HTTP به Client ارسال کند. تا زمانی که Client بتواند فرمت دریافتی را بخواند، می‌تواند از آن استفاده کند. بیشتر Clientها می‌توانند فرمت JSON یا XML را پردازش کنند. به علاوه، Client می‌تواند نوع فرمت درخواستی از Server را با تنظیم مقدار هدر Accept در درخواست ارسالی تعیین کند. اجازه بدهید کار خود را با ایجاد یک مدل ساده که نمایانگر یک محصول است آغاز کنیم.

بر روی پوشه‌ی Models کلیک راست کرده و از منوی Add، گزینه‌ی Class را انتخاب کنید.

نام کلاس را Product گذاشته و کدهای ذیل را در آن بنویسید.

```
namespace HelloWebAPI.Models
{
    public class Product
    {
```

```
        public int Id { get; set; }
        public string Name { get; set; }
        public string Category { get; set; }
        public decimal Price { get; set; }
    }
}
```

مدل ما، چهار Property دارد که در کدهای قبل ملاحظه می‌کنید.

اضافه کردن Controller

در پروژه ای که با استفاده از قالب پیش فرض Web API ایجاد می‌شود، دو Controller نیز به طور خودکار در پروژه‌ی Controller قرار می‌گیرند:

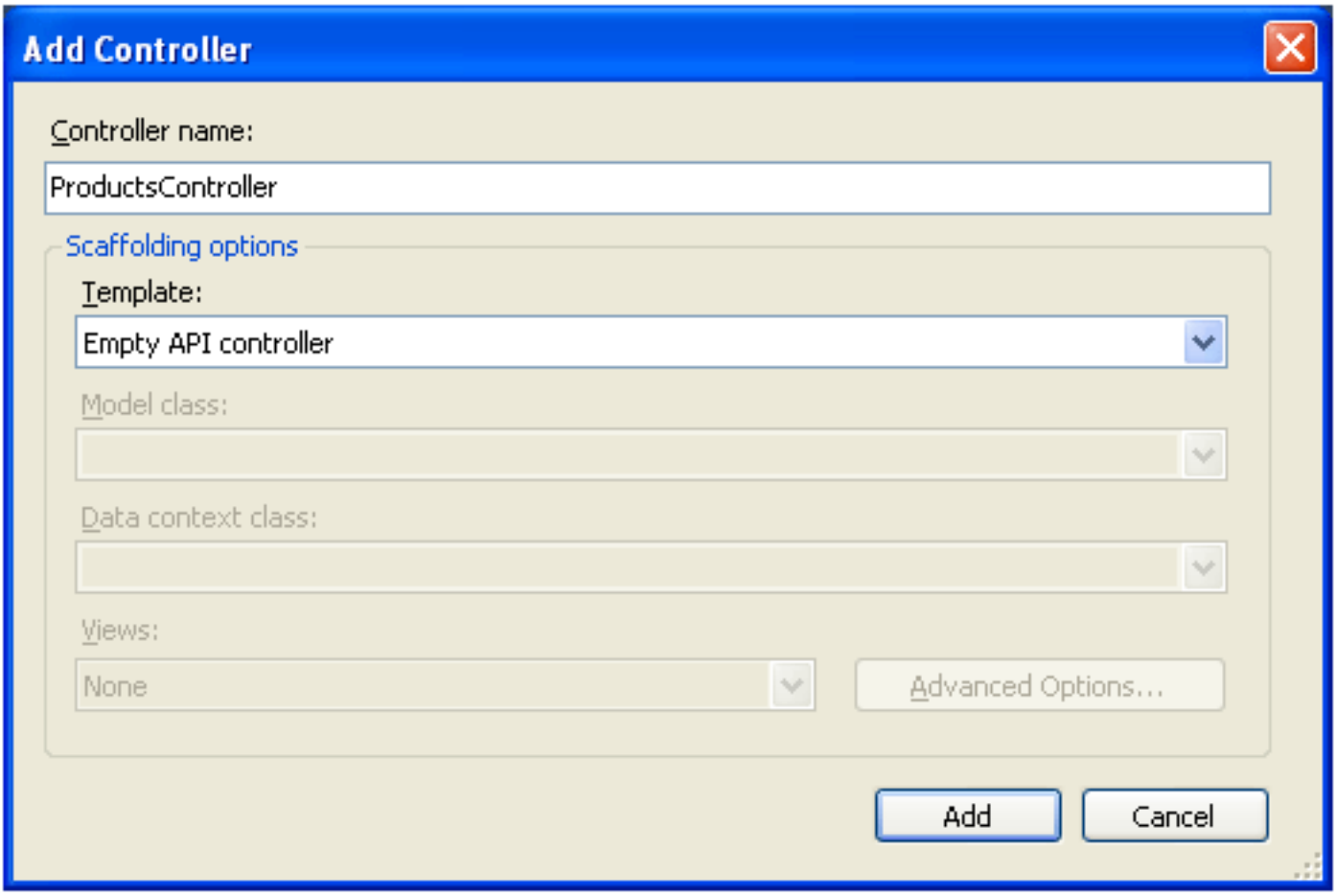
HomeController: یک Controller معمول ASP.NET MVC است که ارتباطی با Web API ندارد.
ValuesController: یک Controller مختص Web API است که به عنوان یک مثال در پروژه قرار داده می‌شود.

توجه: Controllerها در Web API بسیار شبیه به Controllerها در ASP.NET MVC هستند، با این تفاوت که به جای کلاس Controller، از کلاس ApiController ارث می‌برند و بزرگترین تفاوتی که در نگاه اول در متدهای این نوع کلاس‌ها به چشم می‌خورد این است که به جای برگشت Viewها، داده برگشت می‌دهند.

کلاس ValuesController را حذف و یک Controller به پروژه اضافه کنید. بدین منظور، بر روی پوشه‌ی Controllers، کلیک راست کرده و از منوی Add، گزینه‌ی Controller را انتخاب کنید.

توجه: در ASP.NET MVC 4 می‌توانید بر روی هر پوشه‌ی دلخواه در پروژه کلیک راست کرده و از منوی Add، گزینه‌ی Controller را انتخاب کنید. پیشتر فقط با کلیک راست بر روی پوشه‌ی Controller، این گزینه در دسترس بود. حال می‌توان کلاس‌های مرتبط با Controllerهای معمول را در یک پوشه و Controllerهای مربوط به قابلیت Web API را در پوشه‌ی دیگری قرار داد.

نام Controller را ProductsController بگذارید، از قسمت Template، گزینه‌ی Empty API Controller را انتخاب و بر روی دکمه‌ی OK کلیک کنید (شکل ذیل).



فایلی با نام ProductsController.cs در پوشه‌ی Controllers قرار می‌گیرد. آن را باز کنید و کدهای ذیل را در آن قرار دهید.

```
namespace HelloWebAPI.Controllers
{
    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.Net;
    using System.Net.Http;
    using System.Web.Http;
    using HelloWebAPI.Models;

    public class ProductsController : ApiController
    {

        Product[] products = new Product[]
        {
            new Product { Id = 1, Name = "Tomato Soup", Category = "Groceries", Price = 1.39M },
            new Product { Id = 2, Name = "Yo-yo", Category = "Toys", Price = 3.75M },
            new Product { Id = 3, Name = "Hammer", Category = "Hardware", Price = 16.99M }
        };

        public IEnumerable<Product> GetAllProducts()
        {
            return products;
        }

        public Product GetProductById(int id)
        {
            var product = products.FirstOrDefault((p) => p.Id == id);
            if (product == null)
            {
                var resp = new HttpResponseMessage(HttpStatusCode.NotFound);
                throw new HttpResponseException(resp);
            }
            return product;
```

```
        }

        public IEnumerable<Product> GetProductsByCategory(string category)
        {
            return products.Where(
                (p) => string.Equals(p.Category, category,
                    StringComparison.OrdinalIgnoreCase));
        }
    }
}
```

برای ساده نگهداشتن مثال، لیستی از محصولات را در یک آرایه قرار داده ایم اما واضح است که در یک پروژه‌ی واقعی، این لیست از پایگاه داده بازیابی می‌شود. در مورد کلاس‌های HttpResponseMessage و HttpResponseException بعداً توضیح می‌دهیم.
در کدهای Controller قبل، سه متد تعریف شده اند:
متد GetAllProducts که کل محصولات را در قالب نوع IEnumerable<Product> برگشت می‌دهد.
متد GetProductById که یک محصول را با استفاده از مشخصه‌ی آن (خصیصه‌ی Id) برگشت می‌دهد.
متد GetProductsByCategory که تمامی محصولات موجود در یک دسته‌ی خاص را برگشت می‌دهد.

تمام شد! حال شما یک وب سرویس با استفاده از Web API ایجاد کرده اید. هر یک از متدهای قبل در Controller، به یک آدرس به شرح ذیل تناظر دارند.

GetAllProducts به /api/products

GetProductById به /api/products/ *id*

GetProductsByCategory به /api/products/?category= *category*

در آدرس‌های قبل، id و category، مقادیری هستند که همراه با آدرس وارد می‌شوند و در پارامترهای متناظر خود در متدهای مربوطه قرار می‌گیرند. یک Client می‌تواند هر یک از متدها را با ارسال یک درخواست از نوع GET اجرا کند.

در قسمت بعد، کار خود را با تست پروژه و نحوه‌ی تعامل jQuery با آن ادامه می‌دهیم.

نظرات خوانندگان

نویسنده: mze666
تاریخ: ۱۳۹۱/۰۴/۱۳ ۸:۷

سلام آقای راد من MVC رو بلدم ولی کاربرد این WebApi , Web Service رو نمیدونم. یعنی اگر براتون ممکنه چند تا مثال واقعی از این که کجاها استفاده میشه بزنید. ممنون.

نویسنده: بهروز راد
تاریخ: ۱۳۹۱/۰۴/۱۳ ۸:۱۳

وب سرویس‌ها کاربردهای متفاوتی دارند. برای ارتباط بین سیستم ها، استفاده از داده هایی که توسط یک شرکت عرضه میشه مثل اطلاعات آب و هوا یا بورس، عملیات‌های مختلفی که بر روی پایگاه داده انجام میشه، ارسال SMS، تراکنش‌های بانکی و ...

نویسنده: ایمان اسلامی
تاریخ: ۱۳۹۱/۰۴/۱۳ ۸:۱۵

ممنون از مطالب خوبتون
امیدوارم به همین شکل مطلوب ادامه داشته باشه و بهتر از اون ، به زودی شاهد چاپ کتابتون باشیم.

نویسنده: زهرا
تاریخ: ۱۳۹۱/۰۴/۱۳ ۸:۲۳

سلام آقای راد

میخواستم بپرسم که mvc 4 رو چطور به لیست پروژه هام اضافه کنم؟ و اینکه آیا این کاری که شما انجام دادید در asp.net webform هم جواب میده؟ یا اینکه باید در solution یک پروژه mvc 4 ایجاد کرد و از اون استفاده کرد؟

با تشکر

نویسنده: بهروز راد
تاریخ: ۱۳۹۱/۰۴/۱۳ ۹:۲۷

سلام.
اگر نسخه‌ی آفلاین RC اون رو میخواید، از این لینک دریافت کنید.
بله، Web API در ASP.NET Web Forms هم قابل استفاده است.
در پروژه‌های Web Forms، از دیالوگ Add New Item، گزینه‌ی Web API Controller Class رو باید انتخاب کنید. route رو هم باید در متد Application_Start فایل Global.asax به صورت ذیل تعریف کنید.

```
void Application_Start(object sender, EventArgs e)
{
    RouteTable.Routes.MapHttpRoute(
        name: "DefaultApi",
        routeTemplate: "api/{controller}/{id}",
        defaults: new { id = System.Web.Http.RouteParameter.Optional }
        );
}
```

نویسنده: علی

تاریخ: ۱۳۹۱/۰۸/۲۱ ۷:۱۲

سلام آقای راد
نمی دونم چطور می‌شه از آدمایی مثل شما تشکر کرد،مطالب واقعا مفید و آموزندس
خیلی خیلی متشکرم
آقای راد یک سوال از خدمتتون داشتم،مدتیه که من و خانمم در حال ترجمه یک کتاب wcf هستیم ، این اولین کار ترجممونه ،می خواستم ازتون بپرسم که میزان محبوبیت wcf الان تو ایران چقدره ،به نظر شما آینده ای داره ؟ کلا چقدر ارزش وقت گذاشتن داره ؟

نویسنده: محمد صاحب
تاریخ: ۱۳۹۱/۰۸/۲۱ ۸:۴۶

دوست عزیز امیدوارم موفق باشید.
تا آقای راد جواب شما رو بدن این کامنت و قسمت حاشیه این پست رو ببیند بی ارتباط نیست...

نویسنده: بهروز راد
تاریخ: ۱۳۹۱/۰۸/۲۱ ۱۸:۳۴

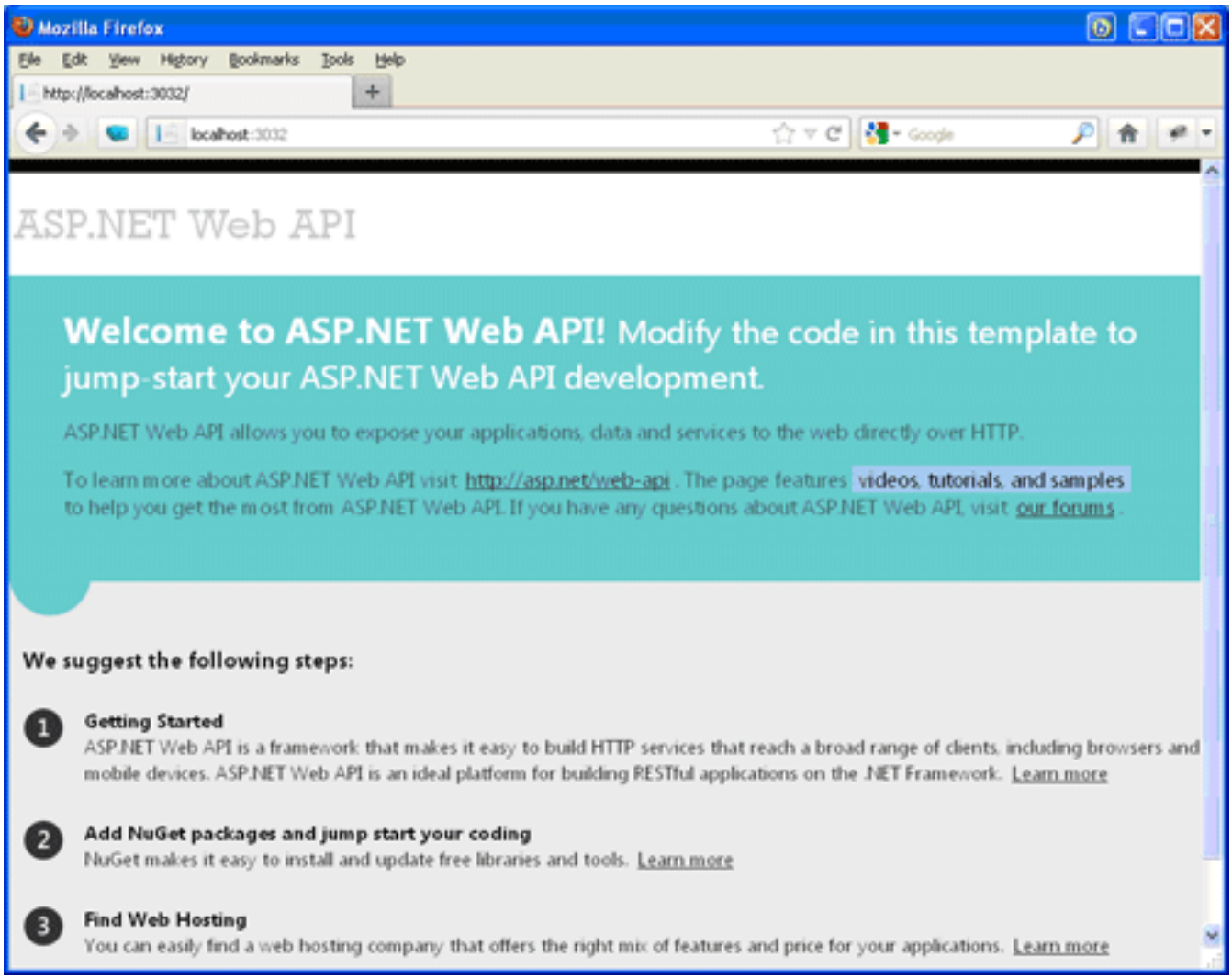
در مورد میزان محبوبیت WCF در ایران اطلاعی ندارم و مطلب خاصی هم در مورد اون در وبلاگ‌های فارسی زبان منتشر نمیشه. اما در حوزه ای که مربوط به خودم هست، حداقل قسمتی از پروژه‌های شرکت فولاد خوزستان که با تیم سازنده‌ی اونها ارتباط دارم از WCF در پروژه‌های اتوماسیون استفاده می‌کنند.
در مورد قسمت دوم سوالتون هم که دوستمون لینک‌های خوبی قرار دادند.

عنوان: ASP.NET Web API - قسمت سوم
نویسنده: بهروز راد
تاریخ: ۱۳۹۱/۰۴/۱۶ ۱۲:۱
آدرس: www.dotnettips.info
گروه‌ها: WCF, ASP.NET Web API, ASMX, Web Service

در قسمت اول به دلایل ایجاد Web API پرداخته شد و در قسمت دوم مثالی ساده از Web API را بررسی کردیم. در این قسمت، مثال قبل را تست کرده و نحوه‌ی تعامل jQuery با آن را بررسی می‌کنیم.

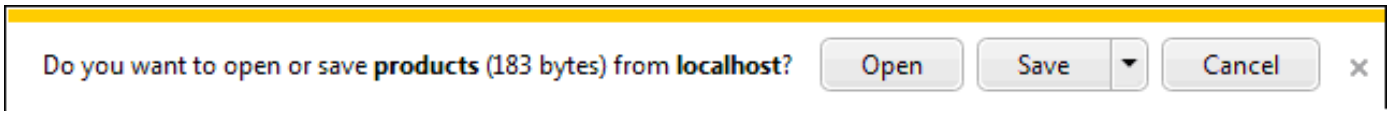
فراخوانی Web API از طریق مرورگر

با فشردن کلید F5، پروژه را اجرا کنید. شکل ذیل ظاهر می‌شود.



صفحه ای که ظاهر می‌شود، یک View است که توسط HomeController و متد Index آن برگشت داده شده است. برای فراخوانی متدهای موجود در کلاس Controller مثال قسمت قبل که مربوط به Web API است، باید به یکی از آدرس‌های اشاره شده در قسمت قبل برویم. به عنوان مثال، برای به دست آوردن لیست تمامی محصولات، به آدرس http://localhost:*xxxx*/api/products بروید. xxxx، شماره‌ی پورتی است که Web Server داخلی Visual Studio در هنگام اجرای پروژه به آن اختصاص می‌دهد. آن را نسبت به پروژه‌ی خود تغییر دهید.
نتیجه‌ی دریافتی بستگی به نوع مرورگری دارد که استفاده می‌کنید. Internet Explorer از شما در مورد باز کردن یا ذخیره‌ی

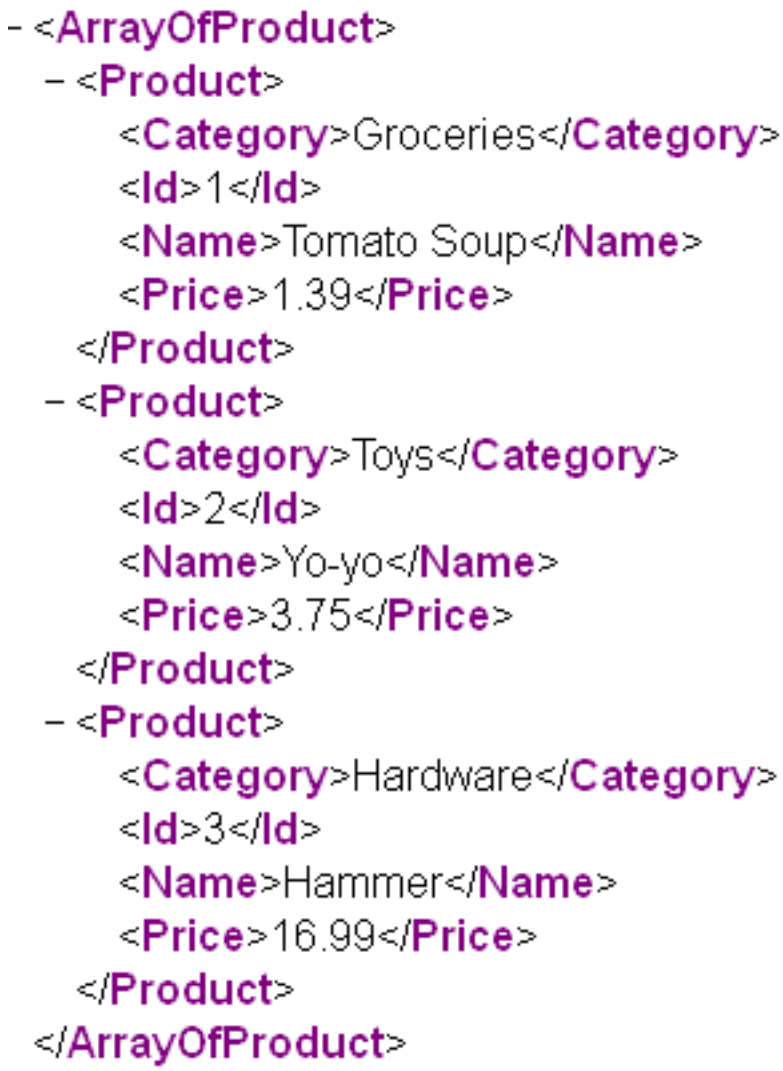
فایلی با نام products پرسش می‌کند (شکل ذیل).



محتوای فایل، بدنه‌ی پاسخ دریافتی است. اگر این فایل را باز کنید، خواهید دید که که محتوای آن، لیستی از محصولات با فرمت JSON مانند ذیل است.

```
[{"Id":1,"Name":"Tomato soup","Category":"Groceries","Price":1.39},{"Id":2,"Name":
"Yo-yo","Category":"Toys","Price":3.75},{"Id":3,"Name":"Hammer","Category":
"Hardware","Price":16.99}]
```

اما مرورگر Firefox، محصولات را در قالب XML نشان می‌دهد (شکل ذیل).



دلیل تفاوت در نتیجه‌ی دریافتی این است که مرورگر Internet Explorer و Firefox، هر یک مقدار متفاوتی را در هدر Accept درخواست، ارسال می‌کنند. بنابراین، Web API نیز مقدار متفاوتی را در پاسخ برگشت می‌دهد.

حال به آدرس‌های ذیل بروید:

http://localhost: *xxxx* /api/products/1
http://localhost: *xxxx* /api/products?category=hardware

اولین آدرس، باید محصولی با مشخصه‌ی 1 را برگشت دهد و دومین آدرس، لیستی از تمامی محصولاتی که در دسته‌ی hardware قرار دارند را برگشت می‌دهد (در مثال ما فقط یک آیتم این شرط را دارد).

نکته: در صورتی که در هنگام فراخوانی هر یک از متدهای Web API با خطای ذیل مواجه شدید، دستور [("AcceptVerbs("GET","POST"] را به ابتدای متدها اضافه کنید.

```
The requested resource does not support http method 'GET'
```

فراخوانی Web API با استفاده از کتابخانه‌ی jQuery

در قسمت قبل، متدهای Web API را مستقیماً از طریق وارد کردن آدرس آنها در نوار آدرس مرورگر فراخوانی کردیم. اما در اکثر اوقات، این متدها با روش‌های برنامه نویسی توسط یک Client فراخوانی می‌شوند. اجازه بدهید Clientیی ایجاد کنیم که با استفاده از jQuery، متدهای ما را فراخوانی می‌کند.
در Solution Explorer، از پوشه‌ی Views و سپس Home، فایل Index.cshtml را باز کنید.

تمامی محتویات این View را حذف و کدهای ذیل را در آن قرار دهید.

```
<!DOCTYPE html>
<html>
<head>
    <title>ASP.NET Web API</title>
    <script src="../../Scripts/jquery-1.7.2.min.js"
        type="text/javascript"></script>
</head>
<body>
    <div>
        <h1>All Products</h1>
        <ul id='products' />
    </div>
    <div>
        <label for="prodId">ID:</label>
        <input type="text" id="prodId" size="5"/>
        <input type="button" value="Search" onclick="find();" />
        <p id="product" />
    </div>
</body>
</html>
```

بازیابی لیستی از محصولات

برای بازیابی لیستی از محصولات، فقط کافی است تا یک درخواست از نوع GET به آدرس "api/products/" بفرستید. این کار با jQuery به صورت ذیل انجام می‌شود.

```
<script type="text/javascript">
    $(document).ready(function () {
        // Send an AJAX request
        $.getJSON("api/products/",
        function (data) {
            // On success, 'data' contains a list of products.
            $.each(data, function (key, val) {

                // Format the text to display.
                var str = val.Name + ': $' + val.Price;

                // Add a list item for the product.
                $('<li/>', { html: str })
                .appendTo($('#products'));
            });
        });
    });
</script>
```

متد getJSON، یک درخواست AJAX از نوع GET را ارسال می‌کند و پاسخ دریافتی آن نیز با فرمت JSON خواهد بود. دومین پارامتر متد getJSON، یک callback است که پس از دریافت موفقیت آمیز پاسخ اجرا می‌شود.

بازیابی یک محصول با استفاده از مشخصه‌ی آن

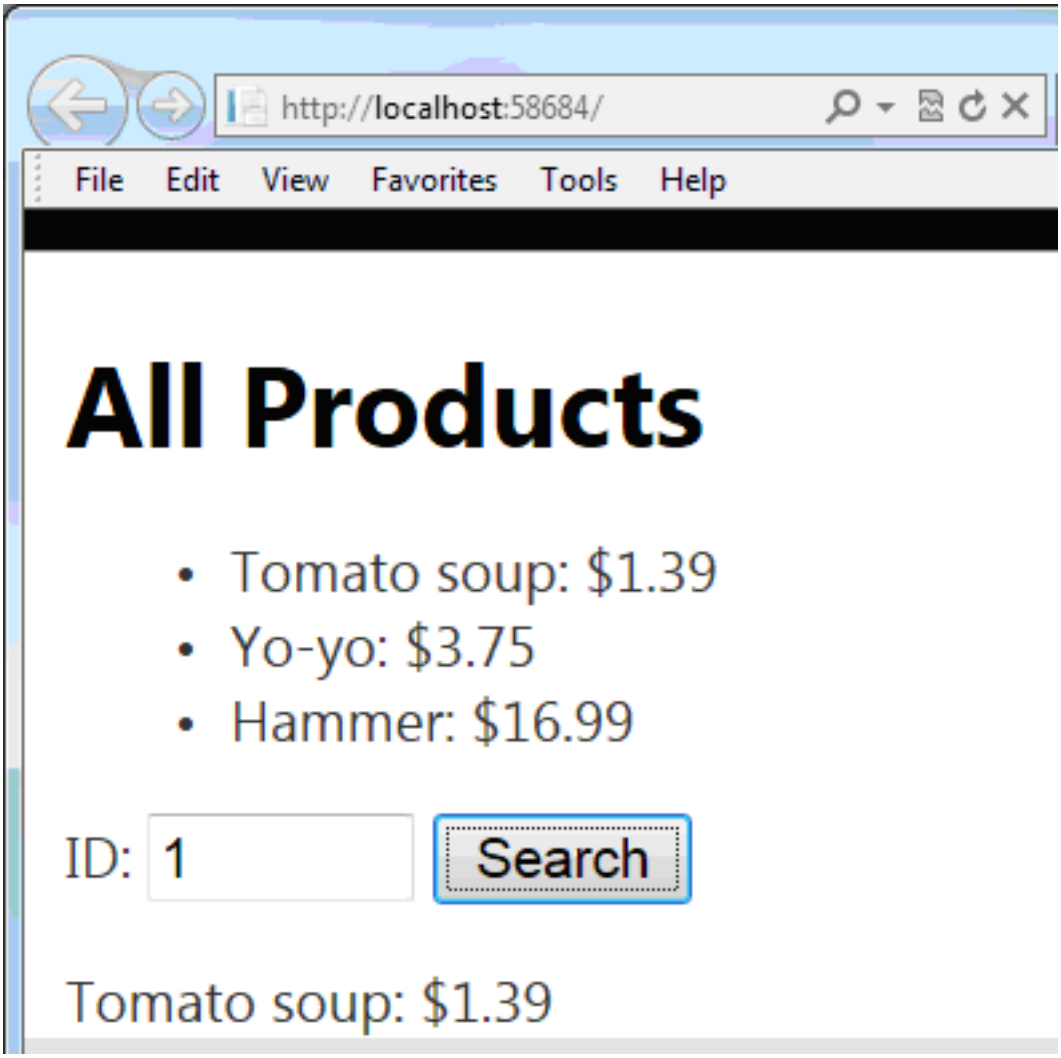
برای بازیابی یک محصول با استفاده از مشخصه‌ی آن، یک درخواست از نوع GET به آدرس "api/products/ *id*" / ارسال کنید. id، مشخصه‌ی محصول است. کد ذیل را در ادامه‌ی کد قبل و پیش از تگ </script> قرار دهید.

```
function find() {
    var id = $('#prodId').val();
    $.getJSON("api/products/" + id,
        function (data) {
            var str = data.Name + ': $' + data.Price;
            $('#product').html(str);
        })
    .fail(
        function (jqXHR, textStatus, err) {
            $('#product').html('Error: ' + err);
        });
}
```

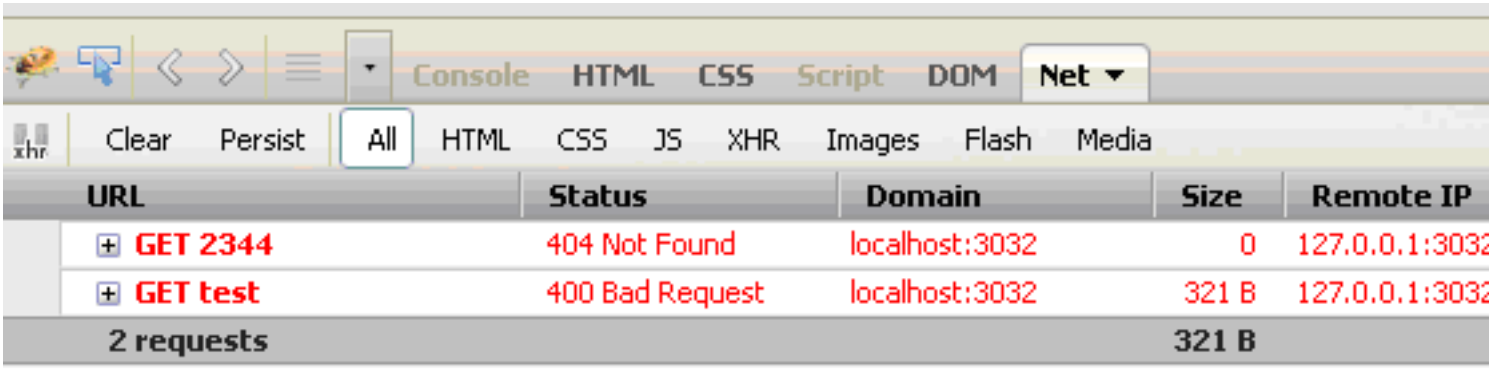
باز هم از متد getJSON استفاده کردیم، اما این بار مقدار id برای آدرس از یک Text Box خوانده و آدرس ایجاد می‌شود. پاسخ دریافتی، یک محصول در قالب JSON است.

اجرای پروژه

پروژه را با فشردن کلید F5 اجرا کنید. پس از نمایش فرم، تمامی محصولات بر روی صفحه نمایش داده می‌شوند. عدد 1 را وارد و بر روی دکمه‌ی Search کلیک کنید، محصولی که مشخصه‌ی آن 1 است نمایش داده می‌شود (شکل ذیل).



اگر مشخصه ای را وارد کنید که وجود ندارد، خطای 404 با مضمون "Error: Not Found" بر روی صفحه نمایش داده می‌شود و در صورتی که به جای عدد، عبارتی غیر عددی وارد کنید، خطای 400 با مضمون: "Error: Bad Request" نمایش داده می‌شود. **در Web API، تمامی پاسخ‌ها باید در قالب کدهای وضعیت HTTP باشند** (شکل ذیل). این یکی از اصول اساسی کار با وب سرویس‌ها است. وفادار ماندن به مفاهیم پایه‌ی وب، دید بهتری در مورد اتفاقاتی که می‌افتد به شما می‌دهد.



در قسمت بعد با مفهوم مسیریابی در ASP.NET Web API آشنا می‌شوید.

نظرات خوانندگان

نویسنده: Nima
تاریخ: ۱۳۹۱/۰۴/۱۶ ۲۰:۶

سلام آقای راد

ممنون از مطلب مفیدتون..سوالی ار حضورتون داشتم ما در وب سرویسهای asmx میتونستیم از sessionها استفاده کنیم تا مثلا اگر میخواستیم از طریق jquery بخواهیم اون وب سرویس رو صدا کنیم این کار فقط برای کاربرانی که در سیستم وارد شده اند امکان پذیر باشد .از لحاظ ملاحظات امنیتی و استفاده از session آیا در قسمتهای بعدی بحث میکنید؟

با تشکر

نویسنده: بهروز راد
تاریخ: ۱۳۹۱/۰۴/۱۷ ۷:۴۶

در Web API در حالت پیش فرض نمی‌تونید از Session استفاده کنید. اصولاً REST اصطلاحاً Stateless هست، اما اگر اصرار به استفاده از Session دارید، باید یک Route Handler سفارشی ایجاد و اینترفیس IRequiresSessionState رو پیاده سازی کنید. سپس پیاده سازی جدید رو به عنوان Route Handler برای route مختص Web API تعریف کنید.
در مورد تصدیق هویت، معمولاً به این شکل عمل میشه که یک فیلتر Authorize سفارشی ایجاد و نام کاربری و کلمه‌ی عبور از طریق یک Header سفارشی به Server ارسال میشه. Web API به خوبی با مفهوم فیلترها در ASP.NET MVC هماهنگ هست. سعی می‌کنم در مطلب جدایی به این موارد بپردازم.

نویسنده: Nima
تاریخ: ۱۳۹۱/۰۴/۱۷ ۹:۵۷

با تشکر از شما آقای راد اگر این زحمت رو بکشین ممنون میشم .دونستن مسائل امنیتی باعث استفاده بهتر از مواردی که شما فرمودین میشه. موفق باشید

نویسنده: مهران کلانتری
تاریخ: ۱۳۹۱/۰۴/۱۷ ۱۸:۵۴

سلام آقای راد خیلی خوب و سلیس توضیح میدید

در رابطه با مسائل امنیتی در این روش خیلی خوب می‌شه اگر توضیحی ارائه بدید.

متشکرم

نویسنده: شهروز جعفری
تاریخ: ۱۳۹۱/۰۴/۱۸ ۲۱:۲۵

در Rest قابلیتی بنام Syndication Feed Fromatter وجود دارد در Web API چطور؟

نویسنده: بهروز راد
تاریخ: ۱۳۹۱/۰۴/۱۹ ۱۰:۳۲

در Web API هم این قابلیت وجود داره.

عنوان: ASP.NET Web API - قسمت چهارم
نویسنده: بهروز راد
تاریخ: ۱۳۹۱/۰۴/۱۹ ۱۱:۱۵
آدرس: www.dotnettips.info
گروه‌ها: WCF, ASP.NET Web API, ASMX, Web Service

آشنایی با مفهوم مسیریابی در Web API

در این قسمت با نحوه‌ی تناظر آدرس‌ها توسط Web API به متدهای موجود در Controller آشنا می‌شوید.
در هر درخواستی که ارسال می‌شود، Web API، انتخاب Controller مناسب را با رجوع به جدولی با نام جدول مسیرها انجام می‌دهد. زمانی که یک پروژه‌ی جدید با استفاده از ASP.NET MVC 4 ایجاد می‌کنید، یک route پیش فرض به صورت ذیل در متد RegisterRoutes قرار می‌گیرد.

```
routes.MapHttpRoute(
    name: "DefaultApi",
    routeTemplate: "api/{controller}/{id}",
    defaults: new { id = RouteParameter.Optional }
);
```

عبارت api، ثابت است و قسمت‌های {controller} و {id} توسط آدرس مقداردهی می‌شوند. زمانی که آدرسی با این الگو تطبیق داشته باشد، کارهای ذیل انجام می‌گیرد:
{controller} به نام Controller تناظر پیدا می‌کند.
نوع درخواست ارسالی (GET، POST، PUT، DELETE) به نام متد تناظر پیدا می‌کند.
اگر قسمت {id} در آدرس وجود داشته باشد، به پارامتر id متد انتخاب شده پاس داده می‌شود.
اگر آدرس دارای Query String باشد، به پارامترهای همنام خود در متد، تناظر پیدا می‌کنند.

در ذیل، مثال هایی را از چند آدرس درخواستی و نتیجه‌ی حاصل از فراخوانی آنها مشاهده می‌کنید.
آدرس api/products/ با نوع درخواست GET به متد GetAllProducts()
آدرس api/products/1 با نوع درخواست GET به متد GetProductById(1)
آدرس api/products?category=hardware با نوع درخواست GET به متد GetProductByCategory("hardware")

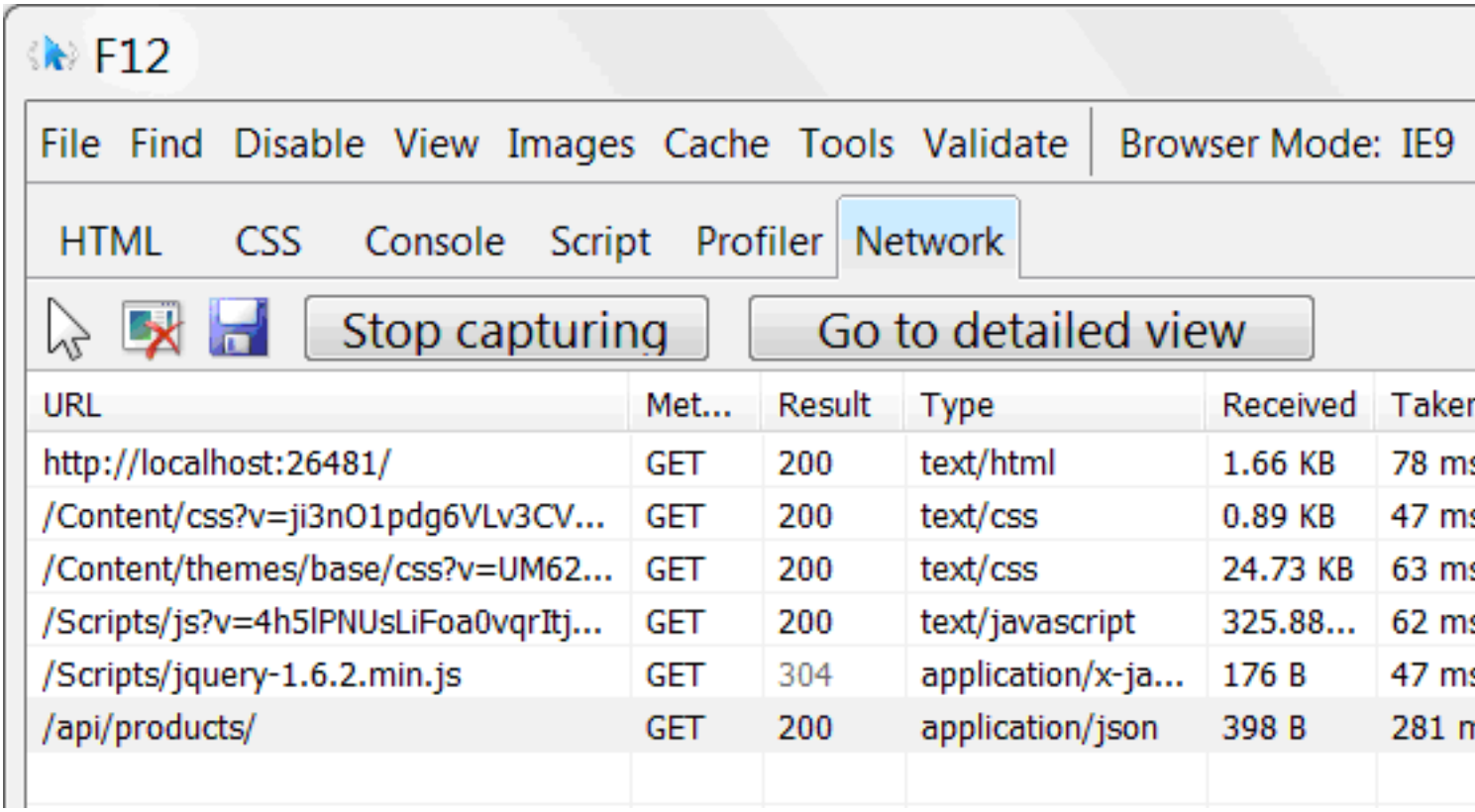
در آدرس اول، عبارت "products" به ProductsController تطبیق پیدا می‌کند. درخواست نیز از نوع GET است، بنابراین Web API به دنبال متدی در Controller می‌گردد که نام آن با عبارت GET "آغاز" شده باشد. همچنین، آدرس شامل قسمت {id} نیز نیست. بنابراین، Web API متدی را انتخاب می‌کند که پارامتر ورودی ندارد. متد GetAllProducts در ProductsController، تمامی این شروط را دارد، پس انتخاب می‌شود.
در دومین آدرس، همان حالت قبل وجود دارد، با این تفاوت که در آدرس درخواستی، قسمت {id} وجود دارد. از آنجا که نوع قسمت {id} در متد GetProductById، int تعریف شده است، باید یک عدد صحیح بعد از آدرس api/products/ وجود داشته باشد تا متد GetProductById فراخوانی شود. این عدد به طور خودکار به نوع int تبدیل شده و در پارامتر اول متد GetProductById قرار می‌گیرد. در ذیل، برخی آدرس‌ها را ملاحظه می‌کنید که معتبر نیستند و باعث بروز خطا می‌شوند.

آدرس api/products/ با نوع درخواست POST، باعث خطای 405Method Not Allowed می‌شود.
آدرس api/users/ با نوع درخواست GET، باعث خطای 404Not Found می‌شود.
آدرس api/products/abc/ با نوع درخواست GET، باعث خطای 400Bad Request می‌شود.

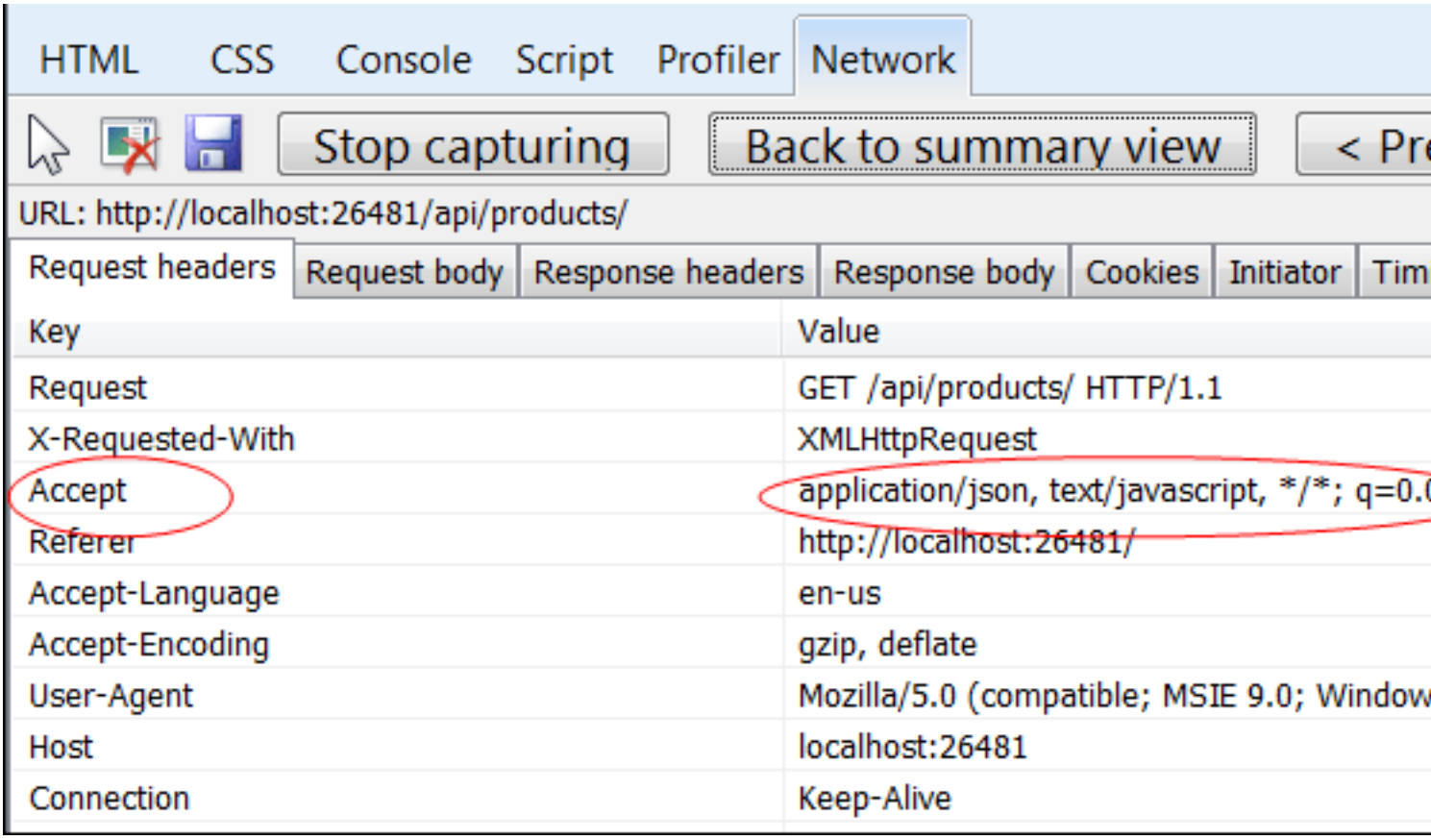
در آدرس اول، Client یک درخواست از نوع POST ارسال کرده است. Web API به دنبال متدی می‌گردد که نام آن با عبارت Post آغاز می‌شود. اما متدی با این شرط در ProductsController وجود ندارد. بنابراین، پاسخی که دریافت می‌شود، عبارت "405 Method Not Allowed" است. درخواست برای آدرس api/users/ نیز معتبر نیست، چون Controllerیی با نام UsersController وجود ندارد. و سومین آدرس نیز بدین دلیل نامعتبر است که قسمت abc نمی‌تواند به یک عدد صحیح تبدیل شود.

مشاهده‌ی درخواست ارسالی و پاسخ دریافتی

زمانی که با یک وب سرویس کار می‌کنید، مشاهده‌ی محتویات درخواست ارسالی و پاسخ دریافتی می‌تواند کاربرد زیادی در درک نحوه‌ی تعامل بین Client و وب سرویس و کشف خطاهای احتمالی داشته باشد. در Firefox با استفاده از افزونه‌ی Firebug و در Internet Explorer 9 به بالا با ابزار Developer Tools آن می‌توان درخواست‌ها و پاسخ‌ها را مشاهده کرد. در Internet Explorer، کلید F12 را برای اجرای Developer Tools فشار دهید. از قسمت Network بر روی دکمه‌ی Start Capturing کلیک کنید. حال کلید F5 را برای بارگذاری مجدد صفحه فشار دهید. Internet Explorer، درخواست و پاسخ رد و بدل شده بین مرورگر و Web Server را مانیتور کرده و گزارشی را نشان می‌دهد (شکل ذیل).



از ستون URL، آدرس /api/products/ را انتخاب و بر روی دکمه‌ی Go to detailed view کلیک کنید. در قسمتی که باز می‌شود، گزینه هایی برای مشاهده‌ی هدرهای درخواست، پاسخ و همچنین بدنه‌ی هر یک وجود دارد. به عنوان مثال، اگر قسمت Request headers را انتخاب کنید، خواهید دید که Internet Explorer از طریق هدر Accept، تقاضای پاسخ در قالب JSON را کرده است (شکل ذیل).



اگر قسمت Response body را انتخاب کنید، پاسخ دریافت شده در قالب JSON را خواهید دید.

در قسمت بعد، با مدیریت کدهای وضعیت HTTP برای اعمال چهارگانه‌ی CRUD آشنا می‌شوید.

نظرات خوانندگان

نویسنده: نیما
تاریخ: ۱۳۹۱/۰۴/۱۹ ۱۲:۱۲

سلام آقای راد

با تشکر از زحمتی که میکشید. فرمودید که :

"بنابراین web api به دنبال متدی در controller می‌گردد که نام آن با عبارت get "آغاز" شده باشد. "

آیا این کار باعث عدم دقت و ایجاد خطاهای ناخواسته نمیشه؟ این فقط متدی با get شروع بشه شاید برای من که خیلی کم mvc کار کردم یکم مشکل دار به نظر برسه.اگر ما دو متد داشته باشیم که در ابتدای آنها get باشد آیا برنامه خطا میگیرد؟ ممنون میشم یکم در این باره توضیح بدین

نویسنده: بهروز راد
تاریخ: ۱۳۹۱/۰۴/۱۹ ۱۲:۲۶

شما محدود به رفتار پیش فرض Web API نیستید. می‌تونید route رو تغییر بدید و نام Action رو هم در اون ذکر کنید.

```
routes.MapHttpRoute(
    name: "DefaultApi",
    routeTemplate: "api/{controller}/{action}/{id}",
    defaults: new { id = RouteParameter.Optional }
);
```

و در ProductsController داشته باشید:

```
[HttpGet]
public string Details(int id)
{
   // do something
}
```

حال درخواستی برای api/products/details/1 باعث اجرای متد Details میشه.
یا حتی می‌تونید route رو تغییر ندید و فقط از [HttpGet] و [HttpPost] و امثال اونها برای تعیین فعل استفاده کنید. به عنوان مثال، اگر route پیش فرض رو تغییر ندید و متد Details رو به شکل قبل داشته باشید، آدرسی مانند api/products/1 با نوع GET باعث میشه تا متد Details اجرا بشه.

نویسنده: نیما
تاریخ: ۱۳۹۱/۰۴/۱۹ ۱۲:۲۷

بسیار ممنونم خیلی مفید بود

نویسنده: آریا
تاریخ: ۱۳۹۱/۰۴/۱۹ ۱۹:۱

خیلی عالی بود. متشکر

نویسنده: رضا.ب
تاریخ: ۱۳۹۱/۰۴/۲۱ ۱۰:۳۰

یه سوال که ربط چندانی به این پست نداره؛
asp.net wep api رو میتونیم یه لایه abstraction حساب کنیم که در اون منطق سیستم(BL) وجود داره و بنابراین از آن در سطح انتزاعی بالاتری در سیستم یا سیستم‌های مشابه استفاده میشن؟ (تاکید سوالم آنجاست که میزان عملکرد موثر asp.net web api تا کجاست؟)
ممنون.

نویسنده: بهروز راد
تاریخ: ۱۳۹۱/۰۴/۲۱ ۱۹:۳۸

شما در سوالتون می‌تونید عبارت "ASP.NET Web API" رو با "Web Service تحت HTTP" جایگزین کنید. در Web Service هم منطق سیستم وجود داره، مثلاً محاسبه‌ی نرخ تورم در یک بازه‌ی زمانی با توجه به 30 قلم کالای اساسی. عملکرد Web API، همان عملکردی است که از یک Web Service تحت HTTP مانند ASMX انتظار دارید.

نویسنده: حمزه ء
تاریخ: ۱۳۹۳/۰۲/۲۰ ۱۲:۲۶

در قسمت سوم آموزش این مثال رو داشتیم :

```
$.getJSON("api/products/"+id,
                function (data) {
                    var str = data.Name + '    ' + data.Price;
                    $('#products').empty();
                    $('#products').html(str);
                }
                );
```

خب تا اینجا api/products/id اجرا میشه .
فرض کنید چند جستجو داریم و نیاز داریم برای هر کدوم اکشن متناظر با اون اجرا بشه برای مثال:
api/products/id
api/products/details/id
حالا چطور میتونم برای دو دکمه تعیین کنم ، با زدن هر کدوم چه تابعی اجرا بشه ؟

بهتر بگم چطور details رو برای یک دکمه به آدرس اضافه کنم ؟

نویسنده: محسن خان
تاریخ: ۱۳۹۳/۰۲/۲۰ ۱۳:۲

از متد click استفاده کنید. داخل callback آن درخواست Ajax ایی را ارسال کنید به سرور.

نویسنده: حمزه ء
تاریخ: ۱۳۹۳/۰۲/۲۰ ۱۸:۲

ممنونم .
-1کدها رو رویداد کلیک نوشتم و اجرا شد . ولی توی آدرس بار مرورگر هیچ تغییری بوجود نیومد ؟ چطور میتونم زمانی که یک متد رو از web api فراخوانی کردم ، همزمان آدرس بار مرورگر هم تغییر کنه ؟
-2 برای اینکه فقط یوزرهای سایت و آنلاین شده یا role های خاص بتونن از اون متد استفاده کنن ، attribute رو بالای اون اضافه کردم ، آیا درسته ؟

```
[Authorize(Roles="Admin")]
        //[Authorize(Users="")]
```

```
        public Product GetProductById(int Id)
        {
            var product = Products.FirstOrDefault(p => p.Id == Id);
            if(product==null)
            {
                throw new HttpResponseException(HttpStatusCode.NotFound);
            }
            return product;
        }
```

نویسنده: محسن خان
تاریخ: ۱۳۹۳/۰۲/۲۰ ۱۸:۵۸

این ASP.NET MVC نیست. ASP.NET Web API است. می‌تونی دستی آدرس خاصی رو در مرورگر وارد کنی و نهایتا مثلا خروجی JSON یا XML بگیری (شاید بهتر باشه یکبار اینکار رو انجام بدی تا حس بهتری نسبت به این فناوری پیدا کنی که کارش چی هست. خروجی‌اش چی هست). در کل هدفش این نیست که خروجی HTML به شما بده. هدفش تامین داده برای کلاینت‌ها هست. سمت کلاینت رو آزاد هستی هر طور که دوست داشتی کار کنی. مثلا یک صفحه‌ی HTML درست کنی و اطلاعات Web API رو بگیری و نمایش بدی.

عنوان: ASP.NET Web API - قسمت پنجم
نویسنده: بهروز راد
تاریخ: ۱۳۹۱/۰۴/۲۴ ۱۳:۱۸
آدرس: www.dotnettips.info
گروه‌ها: WCF, ASP.NET Web API, ASMX, Web Service

مدیریت کدهای وضعیت در Web API

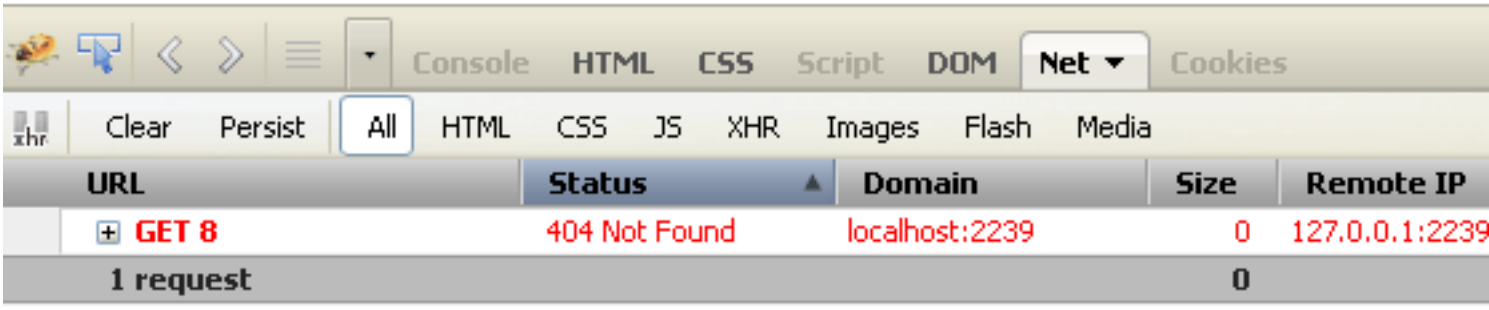
تمامی پاسخ‌های دریافتی از Web API توسط Client، باید در قالب کدهای وضعیت HTTP باشند. دو کلاس جدید با نام‌های HttpResponseMessage و HttpResponseException همراه با ASP.NET MVC 4 معرفی شده اند که ارسال کدهای وضعیت پردازش درخواست به Client را آسان می‌سازند. به عنوان مثال، ارسال وضعیت برای چهار عمل اصلی بازیابی، ایجاد، آپدیت و حذف رکورد را بررسی می‌کنیم.

بازیابی رکورد

بر اساس مستندات پروتوکل HTTP، در صورتی که منبع درخواستی Client پیدا نشد، باید کد وضعیت 404 برگشت داده شود. این حالت را در متد ذیل پیاده سازی کرده ایم.

```
public Product GetProduct(int id)
{
    Product item = repository.Get(id);
    if (item == null)
    {
        throw new HttpResponseException(new HttpResponseMessage(HttpStatusCode.NotFound));
    }
    return item;
}
```

در صورتی که رکوردی با مشخصه‌ی درخواستی پیدا نشد، با استفاده از کلاس HttpResponseException، خطایی به Client ارسال خواهد شد. پارامتر سازنده‌ی این کلاس، شی ای از نوع کلاس HttpResponseMessage است. سازنده‌ی کلاس HttpResponseMessage، مقداری از یک enum با نام HttpStatusCode را می‌پذیرد. مقدار NotFound، نشان از خطای 404 است و زمانی به کار می‌رود که منبع درخواستی وجود نداشته باشد. اگر محصول درخواست شده یافت شد، در قالب JSON برگشت داده می‌شود. در شکل ذیل، پاسخ دریافتی در زمان درخواست محصولی که وجود ندارد را ملاحظه می‌کنید.



ایجاد رکورد

برای ایجاد رکورد، Client درخواستی از نوع POST را همراه با داده‌های رکورد در بدنه‌ی درخواست به Server ارسال می‌کند. در ذیل، پیاده سازی ساده ای از این حالت را مشاهده می‌کنید.

```
public Product PostProduct(Product item)
{
    item = repository.Add(item);
    return item;
}
```

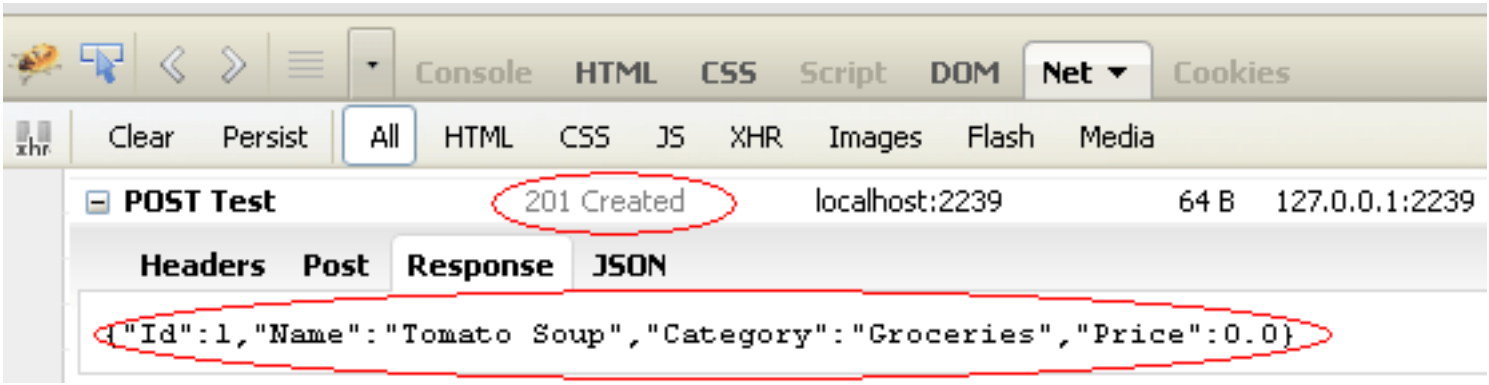
این پیاده سازی کار می‌کند اما کمبودهایی دارد:

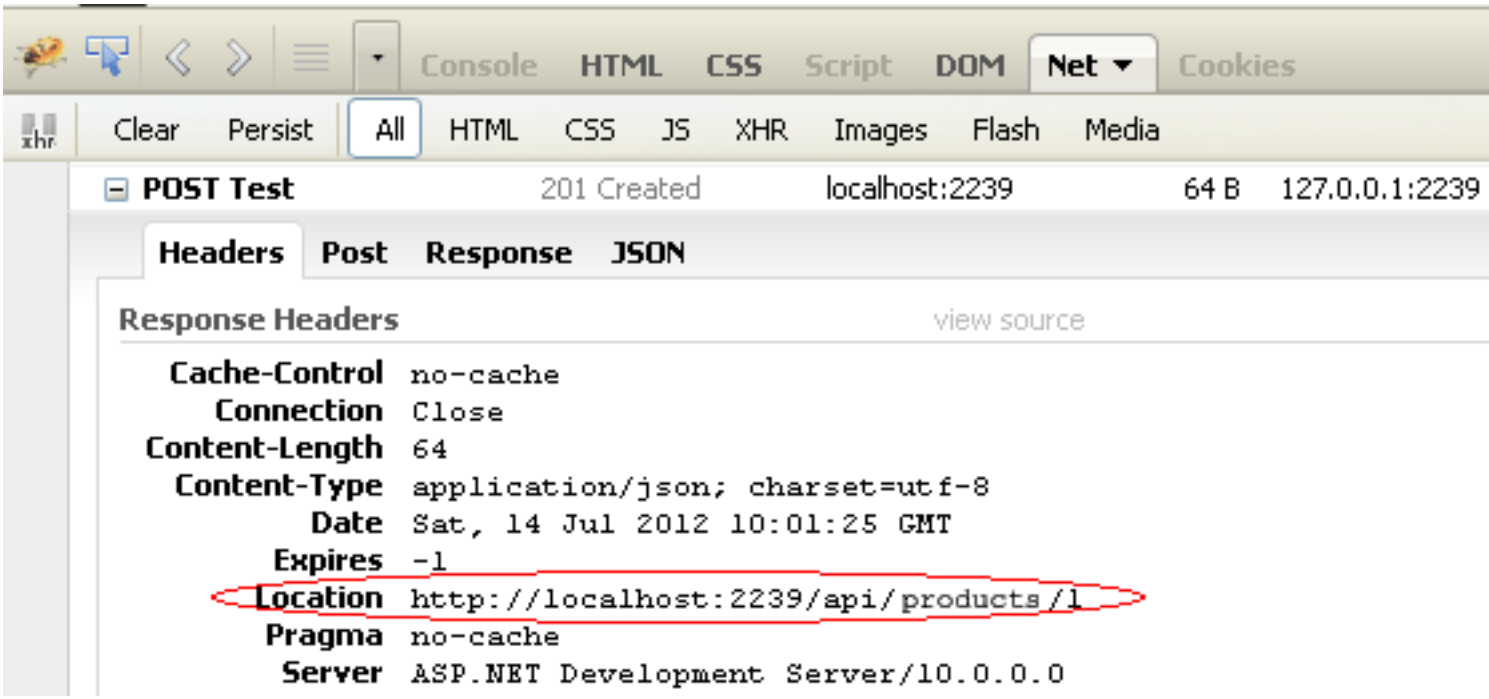
کد وضعیت پردازش درخواست : به طور پیش فرض، Web API، کد 200 را در پاسخ ارسال می‌کند، اما بر اساس مستندات پروتوکل HTTP، زمانی که یک درخواست از نوع POST منجر به تولید منبعی می-شود، Server باید کد وضعیت 201 را به Client برگشت بدهد.
آدرس منبع جدید ایجاد شده : بر اساس مستندات پروتوکل HTTP، زمانی که منبعی بر روی Server ایجاد می‌شود، باید آدرس منبع جدید ایجاد شده از طریق هدر Location به Client ارسال شود.
با توجه به این توضیحات، متد قبل به صورت ذیل در خواهد آمد.

```
public HttpResponseMessage PostProduct(Product item)
{
    item = repository.Add(item);
    var response = Request.CreateResponse(HttpStatusCode.Created, item);

    string uri = Url.Link("DefaultApi", new { id = item.Id });
    response.Headers.Location = new Uri(uri);
    return response;
}
```

همان طور که ملاحظه می‌کنید، خروجی متد از نوع کلاس HttpResponseMessage است، چون با استفاده از این نوع می‌توانیم جزئیات مورد نیاز را در مورد نتیجه‌ی پردازش درخواست به مرورگر ارسال کنیم. همچنین، داده‌های رکورد جدید نیز در بدنه‌ی پاسخ، با یک فرمت مناسب مانند XML یا JSON برگشت داده می‌شوند. با استفاده از متد CreateResponse کلاس Request و پاس دادن کد وضعیت و شی ای که قصد داریم به Client ارسال شود به این متد، شی ای از نوع کلاس HttpResponseMessage ایجاد می‌کنیم. آدرس منبع جدید نیز با استفاده از response.Headers.Location مشخص شده است. نمونه ای از پاسخ دریافت شده در سمت Client به صورت ذیل است.





آپدیت رکورد

آپدیت با استفاده از درخواست‌های از نوع PUT انجام می‌گیرد. یک مثال ساده در این مورد.

```
public void PutProduct(int id, Product product)
{
    product.Id = id;
    if (!repository.Update(product))
    {
        throw new HttpResponseException(new HttpResponseMessage(HttpStatusCode.NotFound));
    }
}
```

نام متد با عبارت Put آغاز شده است. بنابراین توسط Web API برای پردازش درخواست‌های از نوع PUT در نظر گرفته می‌شود. متد قبل، دو پارامتر ورودی دارد. id برای مشخصه‌ی محصول، و محصول آپدیت شده که در پارامتر دوم قرار می‌گیرد. مقدار پارامتر id از آدرس دریافت می‌شود و مقدار پارامتر product از بدنه‌ی درخواست. به طور پیش فرض، Web API، مقدار داده هایی با نوع ساده مانند string، int و bool را از طریق route، و مقدار نوع‌های پیچیده‌تر مانند داده‌های یک کلاس را از بدنه‌ی درخواست می‌خواند.

حذف یک رکورد

حذف یک رکورد، با استفاده از درخواست‌های از نوع DELETE انجام می‌گیرد. یک مثال ساده در این مورد.

```
public HttpResponseMessage DeleteProduct(int id)
{
    repository.Remove(id);
    return new HttpResponseMessage(HttpStatusCode.NoContent);
}
```

بر اساس مستندات پروتکل HTTP، اگر منبعی که Client قصد حذف آن را دارد از پیش حذف شده است، نباید خطایی به وی گزارش شود. معمولاً در متدهایی که وظیفه‌ی حذف منبع را بر عهده دارند، کد 204 مبنی بر پردازش کامل درخواست و پاسخ خالی برگشت داده می‌شود. این کد با استفاد از مقدار NoContent برای HttpStatusCode مشخص می‌شود.

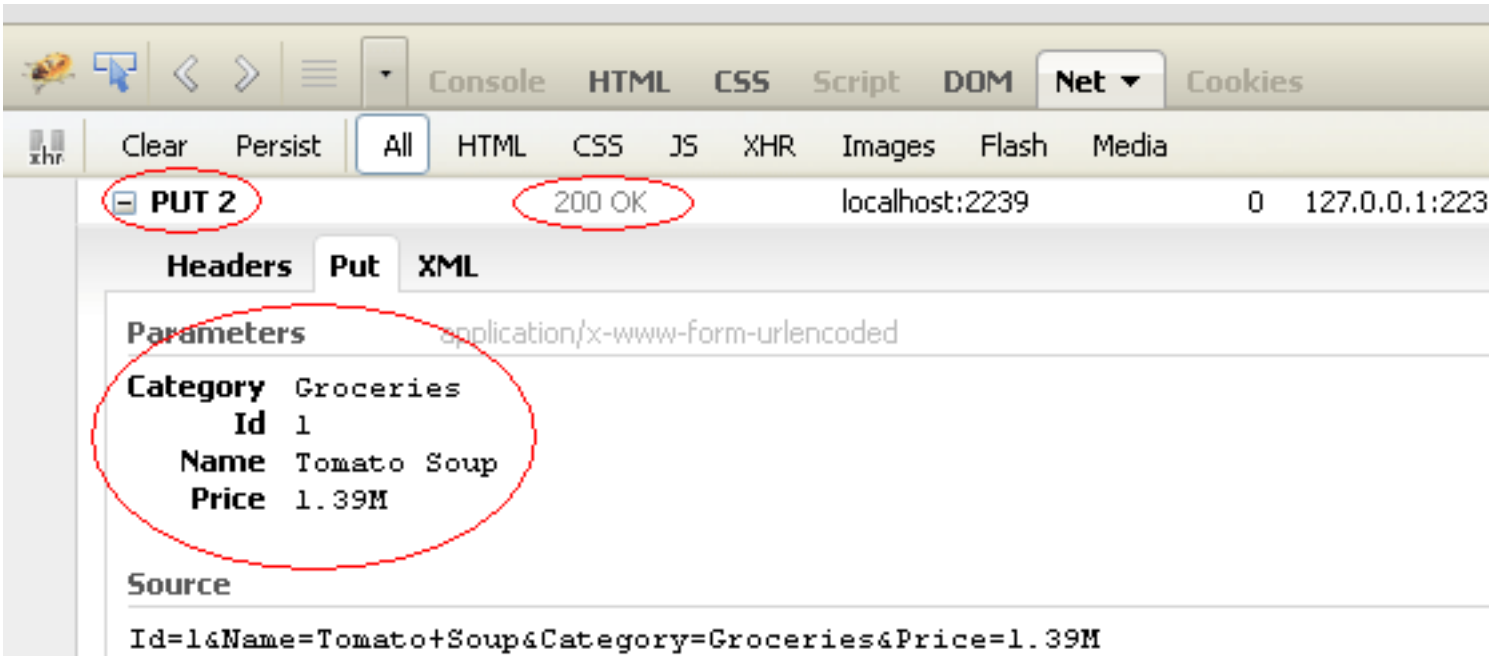
فراخوانی متدها و مدیریت کدهای وضعیت HTTP در سمت Client

حال ببینیم چگونه می‌توان از متدهای قبل در سمت Client استفاده و خطاهای احتمالی آنها را مدیریت کرد.
بهتر است مثال را برای حالتی که در آن رکوردی آپدیت می‌شود بررسی کنیم. کدهای مورد نیاز برای فراخوانی متد PutProduct در سمت Client به صورت ذیل است.

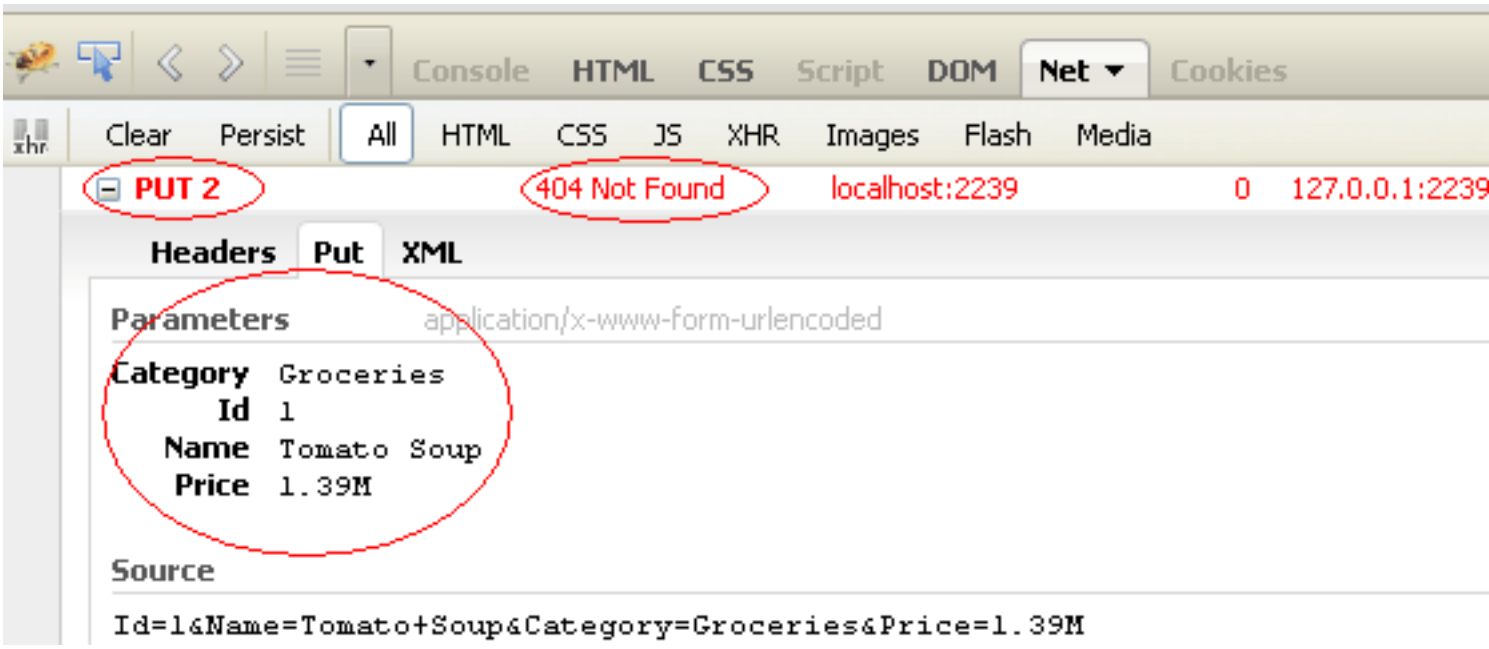
```
var id = $("#myTextBox").val();

$.ajax({
  url: "/api/Test/" + id,
   type: 'PUT',
   data: { Id: "1", Name: "Tomato Soup", Category: "Groceries", Price: "1.39M" },
   cache: false,
   statusCode: {
       200: function (data) {
           alert("آپدیت انجام شد");
        },
       404:
            function () {
                alert("خطا در آپدیت");
            }
     }
});
```

از متدهای get، getJson یا post در jQuery نمی‌توان برای عمل آپدیت استفاده نمود، چون Web API انتظار دارد تا نام فعل درخواستی، PUT باشد. اما با استفاده از متد ajax و ذکر نام فعل در پارامتر type آن می‌توان نوع درخواست را PUT تعریف کرد.
خط 5 بدین منظور است. از طریق خصیصه‌ی statusCode نیز می‌توان کدهای وضعیت مختلف HTTP را بررسی کرد. دو کد 200 و 404 که به ترتیب نشان از موفقیت و عدم موفقیت در آپدیت رکورد هستند تعریف شده و پیغام مناسب به کاربر نمایش داده می‌شود.
در حالتی که آپدیت با موفقیت همراه باشد، بدنه‌ی پاسخ به شکل ذیل است.



و در صورتی که خطایی رخ دهد، بدنه‌ی پاسخ دریافتی به صورت ذیل خواهد بود.


Console
HTML
CSS
Script
DOM
Net
Cookies
Clear
Persist
All
HTML
CSS
JS
XHR
Images
Flash
Media
PUT 2
404 Not Found
localhost:2239
0
127.0.0.1:2239
Headers
Put
XML
Parameters
application/x-www-form-urlencoded
Category Groceries
Id 1
Name Tomato Soup
Price 1.39M
Source
Id=1&Name=Tomato+Soup&Category=Groceries&Price=1.39M

نظرات خوانندگان

نویسنده: princedotnet
تاریخ: ۱۳۹۱/۰۴/۳۱ ۰:۳۰

سلام جناب راد
2 تا سوال داشتم :
1.چطور می‌تونم اطلاعات گرفته شده از WebAPI رو توسط JSON.NET در یک پروژه سیلورلایت Deserialize کنم؟
2.چطور مدل هایی که در اون از روابط many to many - many to one یا... در Entity استفاده شده رو از یک WebAPI بگیرم؟
ممنون

نویسنده: آزاده
تاریخ: ۱۳۹۲/۰۶/۱۹ ۱۱:۴۶

سلام، با تشکر؛ من در صورتی که بخواهم کاری کنم که کاربر فقط از توی فرم و از طریق jqueyهای نوشته شده بتونه به اطلاعات دسترسی داشته باشد، یعنی در صورتی که از آدرس بار بروزر استفاده کرد، خروجی رو نگیرد چیکار باید بکنم؟

ممنون

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۶/۱۹ ۱۲:۸

از محدودیت POST استفاده کنید بجای GET.

نویسنده: آزاده
تاریخ: ۱۳۹۲/۰۶/۱۹ ۱۳:۱۲

سلام . ممنون از راهنماییتون.
یعنی همون متدی که دارم رو فقط به نوع Post تغییر بدم کافیه. و از اون به بعد از آدرس بار نمی‌شه بهش دسترسی داشت.
احتیاجی به تنظیمات خاصی نداره دیگه؟

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۶/۱۹ ۱۳:۳۴

کار معمولی با یک آدرس در مرورگر یعنی حالت Get. میشه این رو تغییر داد به Post که با بازکردن ساده آدرس در مرورگر کار نکنه.