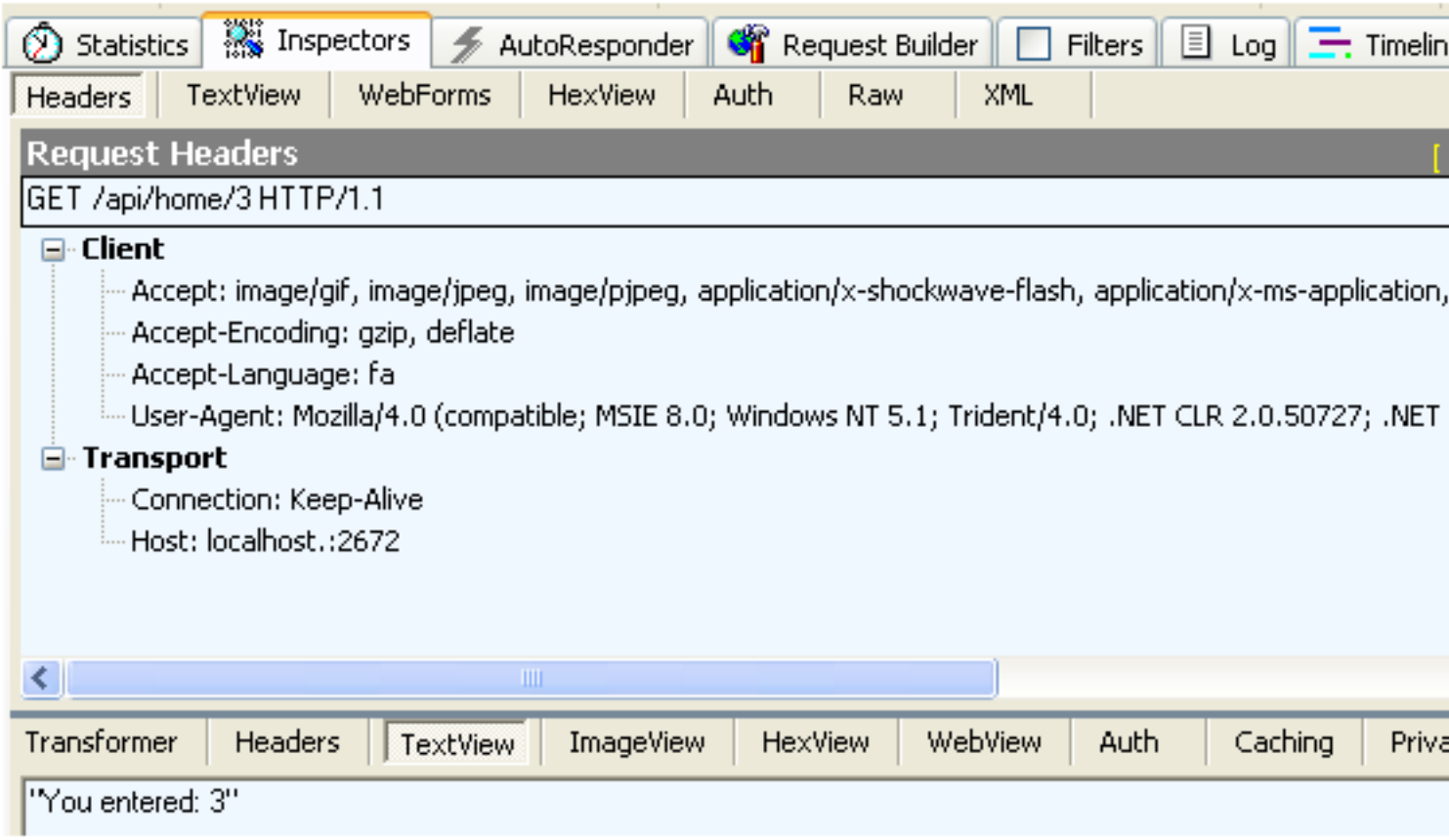
عنوان: **ASP.NET Web API - قسمت دوم**
نویسنده: بهروز راد
تاریخ: ۱۳۹۱/۰۴/۱۳ ۷:۵۸
آدرس: www.dotnettips.info
برچسب‌ها: WCF, ASP.NET Web API, ASMX, Web Service

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 أفعال 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. أعمال مورد نظر در قالب متدها در Controller تعریف می‌شوند. در مثال قبل، متد Get، یکی از أعمال 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.



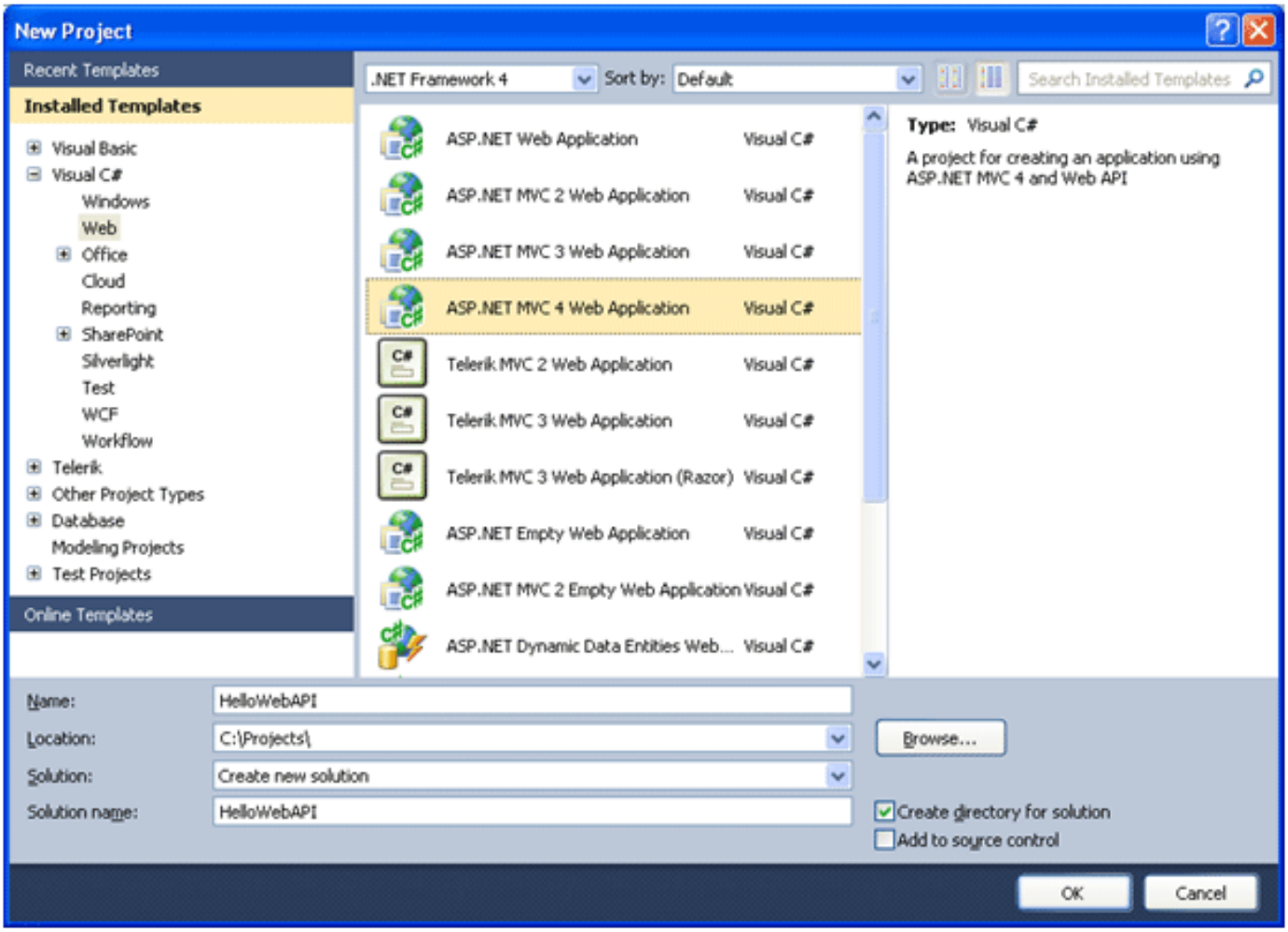
در اینجا نتیجه، عبارت "You entered: 3" است که به صورت یک متن ساده برگشت داده شده است.

ایجاد یک پروژه‌ی Web API

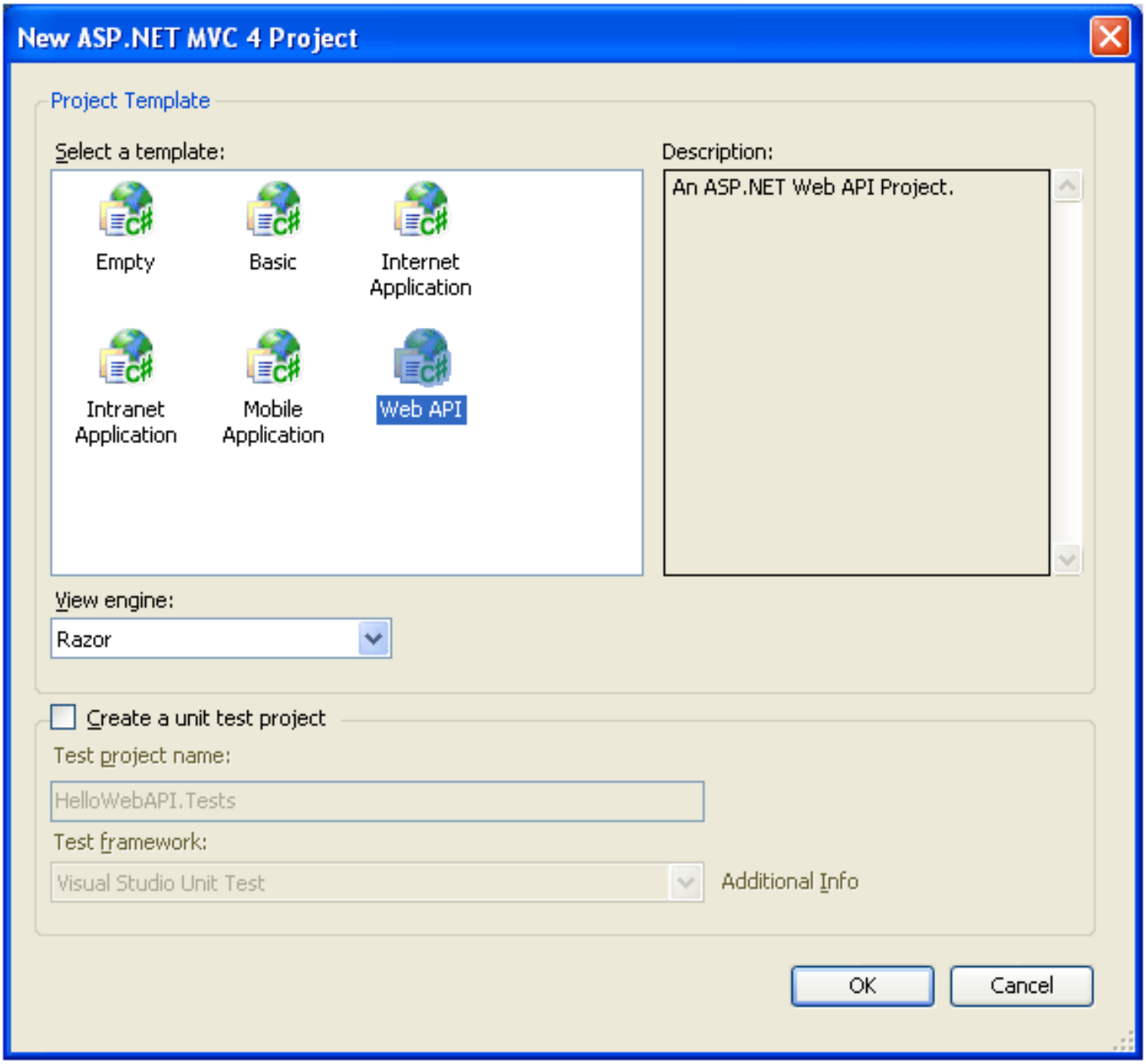
در Visual Studio، مسیر ذیل را طی کنید.

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:

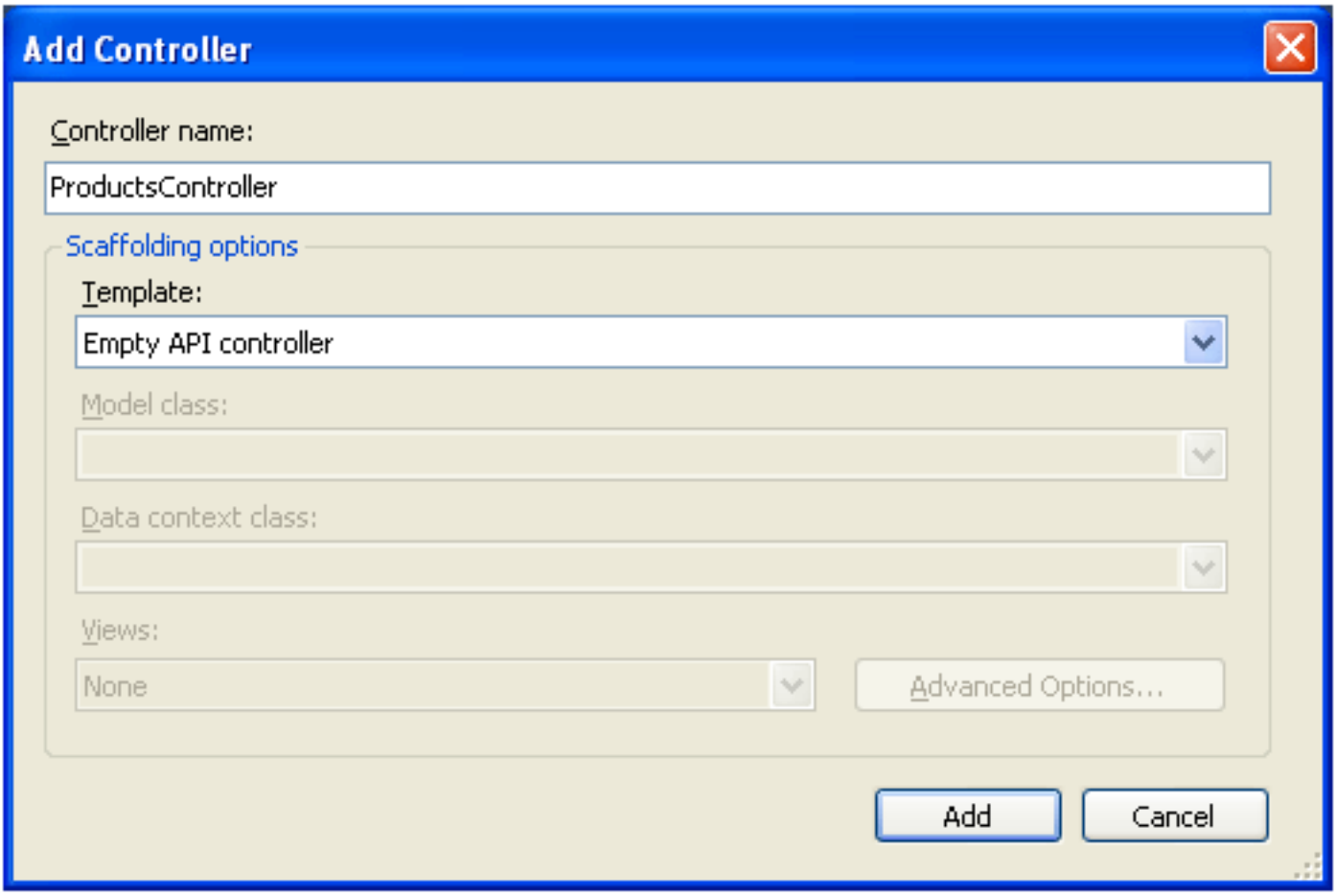
HomeController: یک Controller معمول ASP.NET MVC است که ارتباطی با Web API ندارد.
ValuesController: یک Controller مختص Web API است که به عنوان یک مثال در پروژه قرار داده می‌شود.

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.