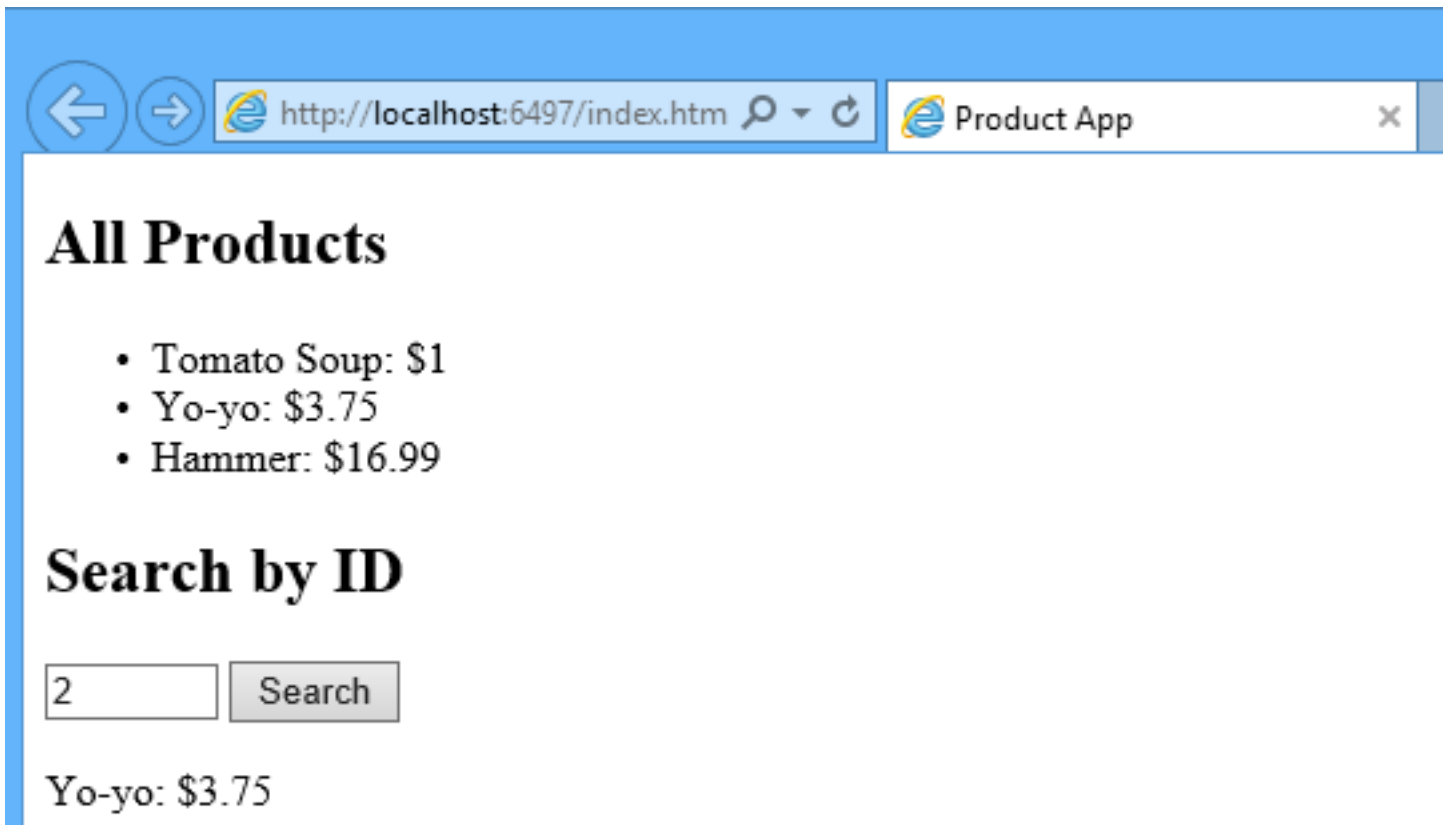


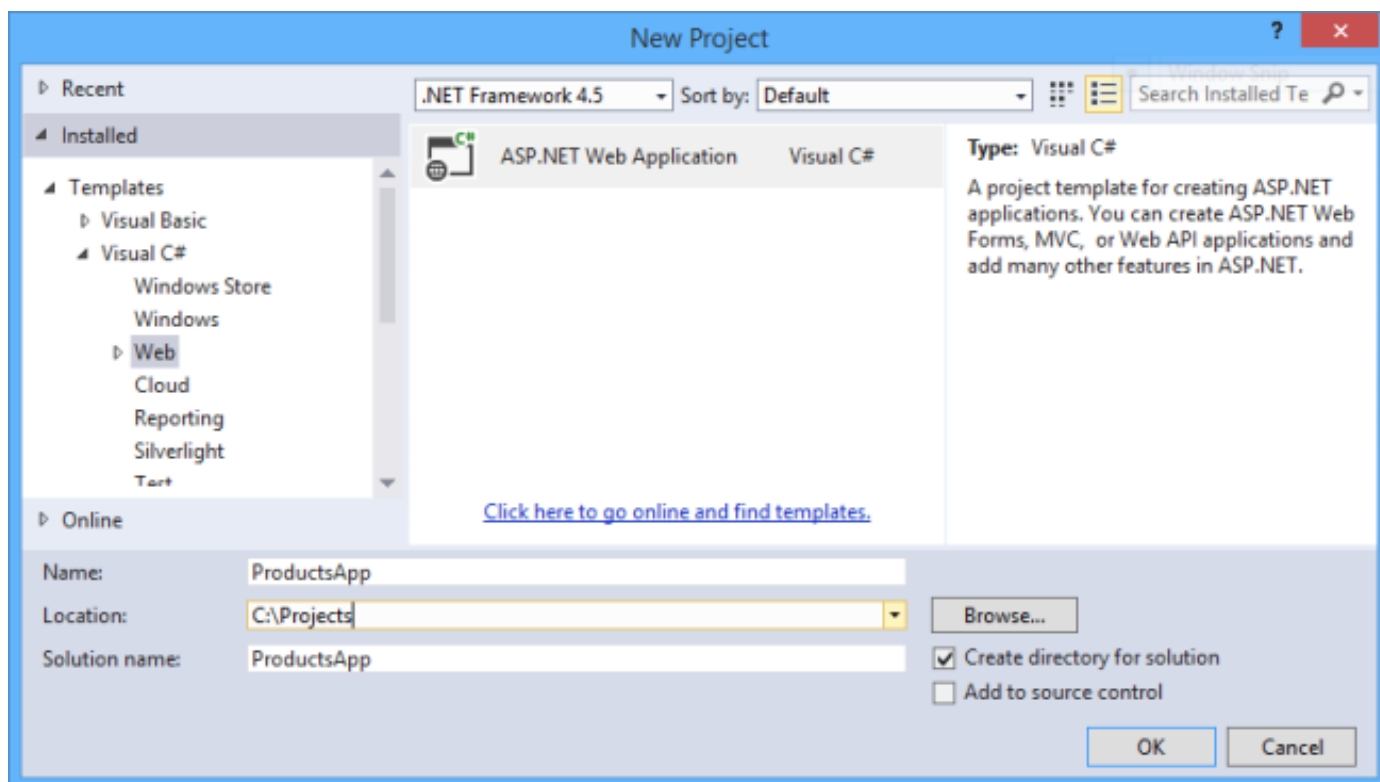
HTTP تنها برای به خدمت گرفتن صفحات وب نیست. این پروتکل همچنین پلتفرمی قدرتمند برای ساختن API هایی است که سرویس‌ها و داده را در دسترس قرار می‌دهند. این پروتکل ساده، انعطاف پذیر و در همه جا حاضر است. هر پلتفرمی که فکرش را بتوانید بکنید کتابخانه ای برای HTTP دارد، بنابراین سرویس‌های HTTP می‌توانند بازه بسیار گسترده ای از کلاینت‌ها را پوشش دهند، مانند مرورگرها، دستگاه‌های موبایل و اپلیکیشن‌های مرسوم دسکتاپ.

ASP.NET Web API فریم ورکی برای ساختن API های وب بر روی فریم ورک دات نت است. در این مقاله با استفاده از این فریم ورک، API وبی خواهیم ساخت که لیستی از محصولات را بر می‌گرداند. صفحه وب کلاینت، با استفاده از jQuery نتایج را نمایش خواهد داد.

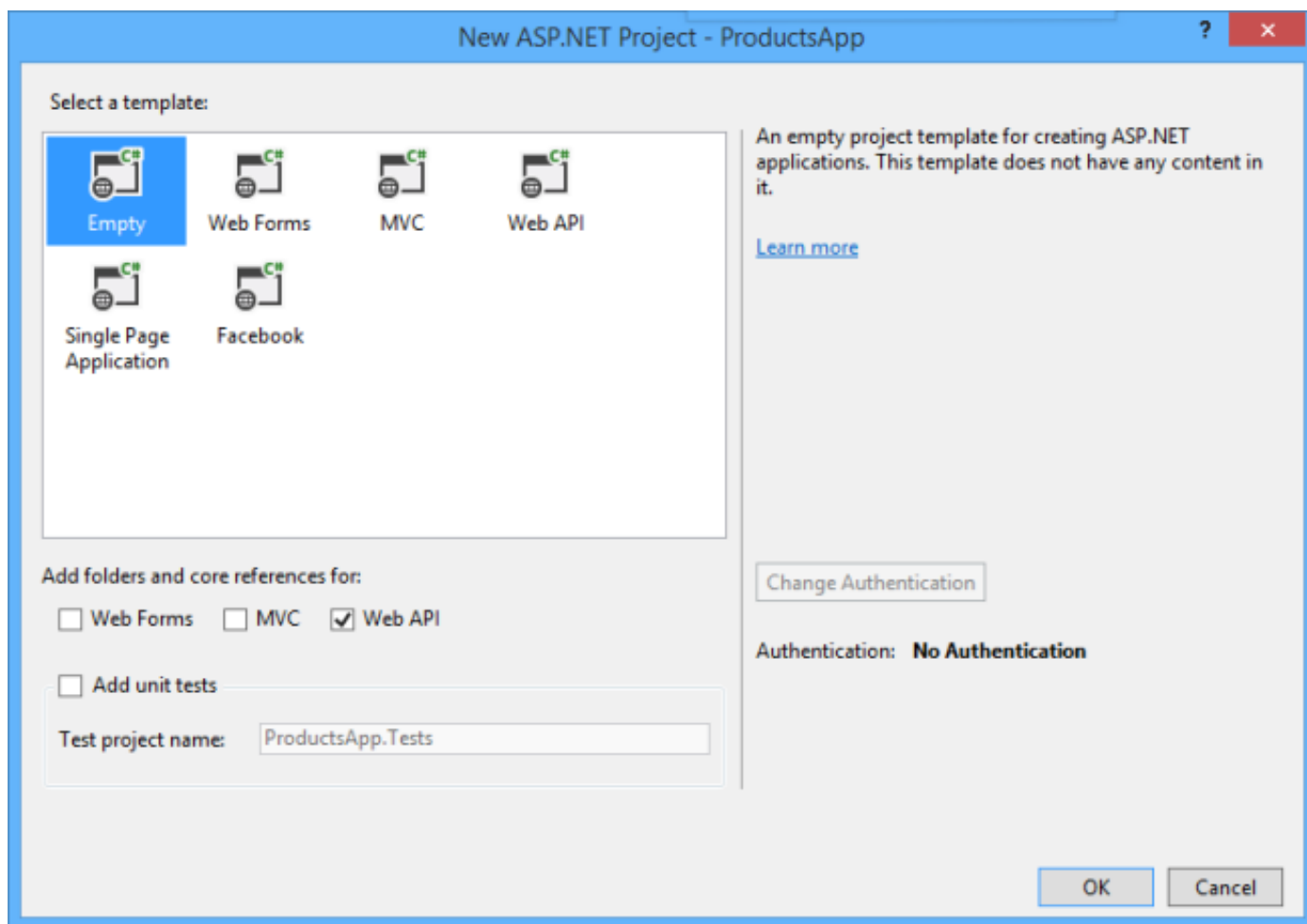


یک پروژه Web API بسازید

در ویژوال استودیو 2013 پروژه جدیدی از نوع ASP.NET Web Application بسازید و نام آن را "ProductsApp" انتخاب کنید.



در دیالوگ New ASP.NET Project قالب Empty را انتخاب کنید و در قسمت "Add folders and core references for" گزینه Web API را انتخاب نمایید.



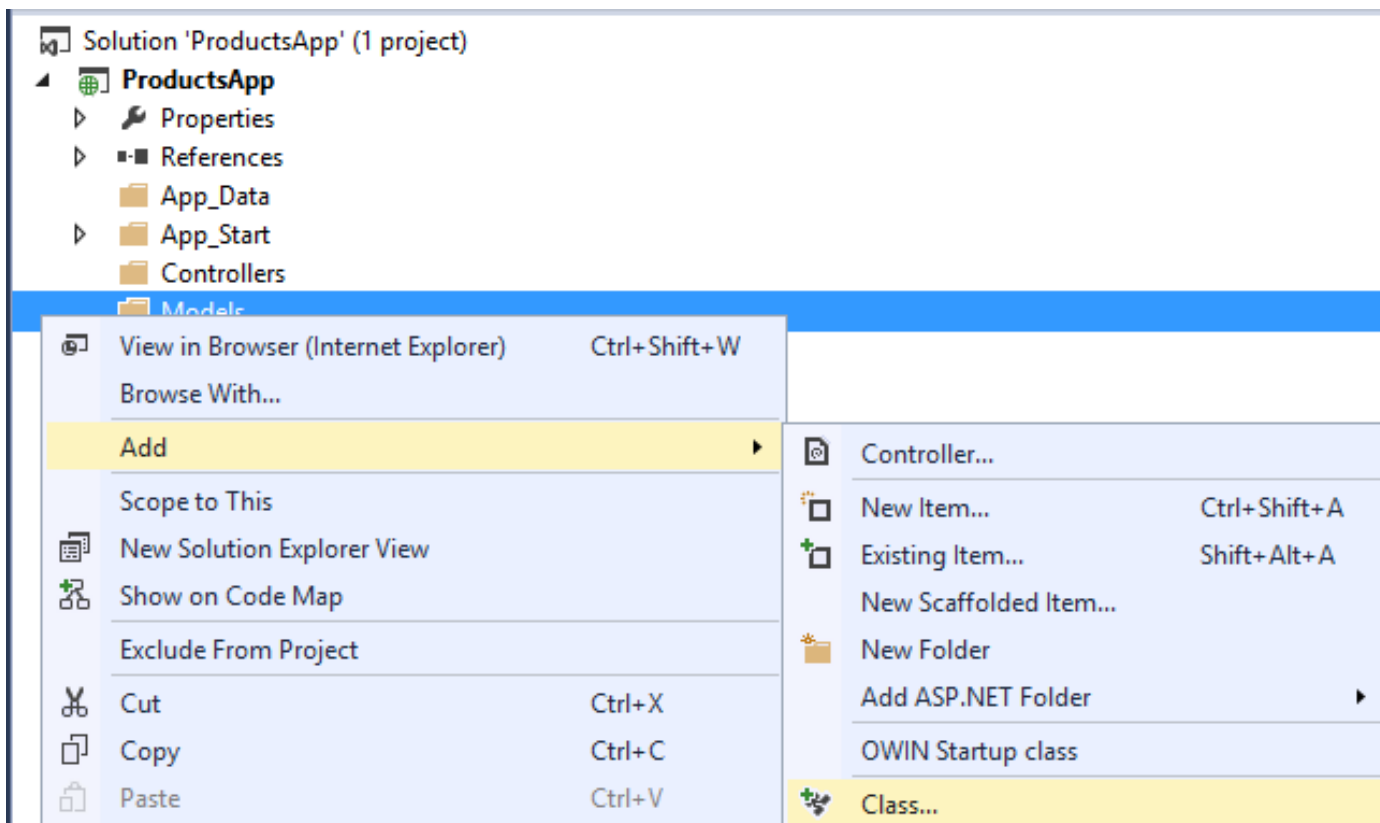
می توانید از قالب Web API هم استفاده کنید. این قالب با استفاده از ASP.NET MVC صفحات راهنمای API را خواهد ساخت. در این مقاله از قالب Empty استفاده میکنیم تا تمرکز اصلی، روی خود فریم ورک Web API باشد. بطور کلی برای استفاده از این فریم ورک لازم نیست با ASP.NET MVC آشنایی داشته باشید.

افزودن یک مدل

یک مدل (model) آبجکتی است که داده اپلیکیشن شما را معرفی می کند. ASP.NET Web API می تواند بصورت خودکار مدل شما را به JSON, XML و برخی فرمت های دیگر مرتب (serialize) کند، و سپس داده مرتب شده را در بدنه پیام HTTP Response بنویسد. تا وقتی که یک کلاینت بتواند فرمت مرتب سازی داده ها را بخواند، می تواند آبجکت شما را deserialize کند. اکثر کلاینت ها می توانند XML یا JSON را تفسیر کنند. بعلاوه کلاینت ها می توانند فرمت مورد نظرشان را با تنظیم Accept header در پیام HTTP Request مشخص کنند.

بگذارید تا با ساختن مدلی ساده که یک محصول (product) را معرفی میکند شروع کنیم.

کلاس جدیدی در پوشه Models ایجاد کنید.



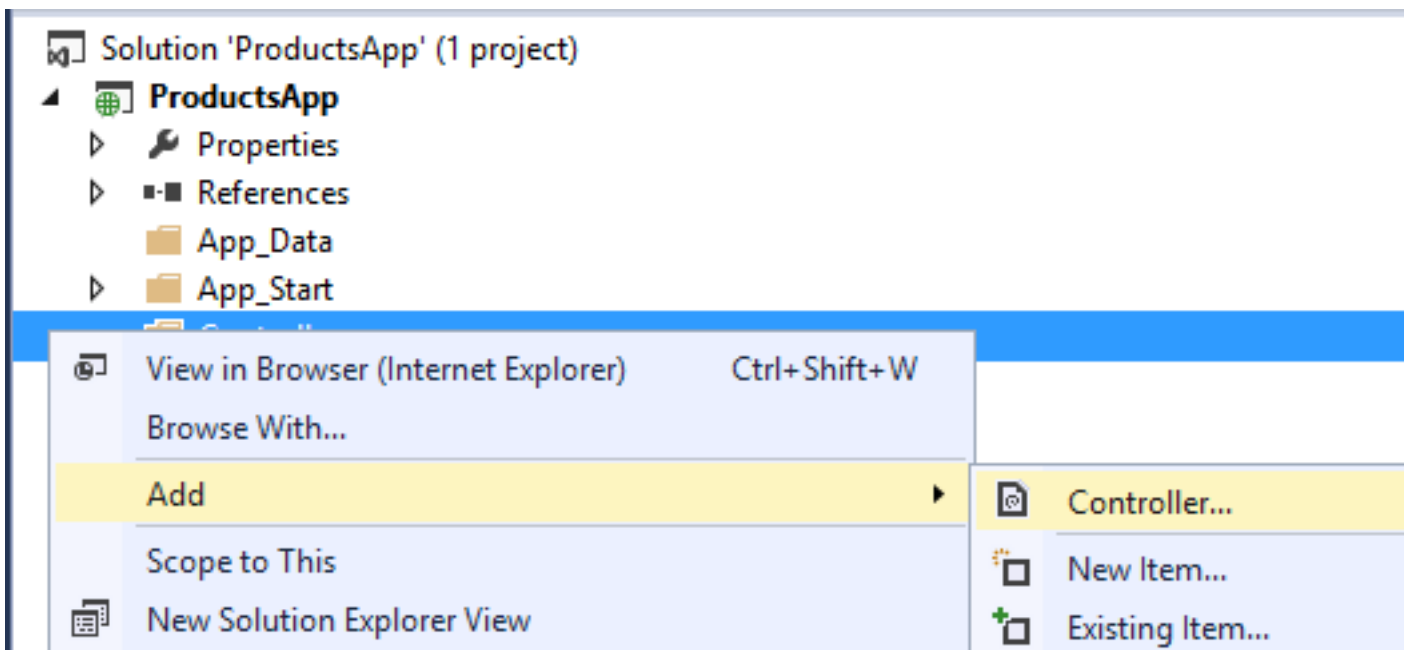
نام کلاس را به "Product" تغییر دهید، و خواص زیر را به آن اضافه کنید.

```
namespace ProductsApp.Models
{
    public class Product
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Category { get; set; }
        public decimal Price { get; set; }
    }
}
```

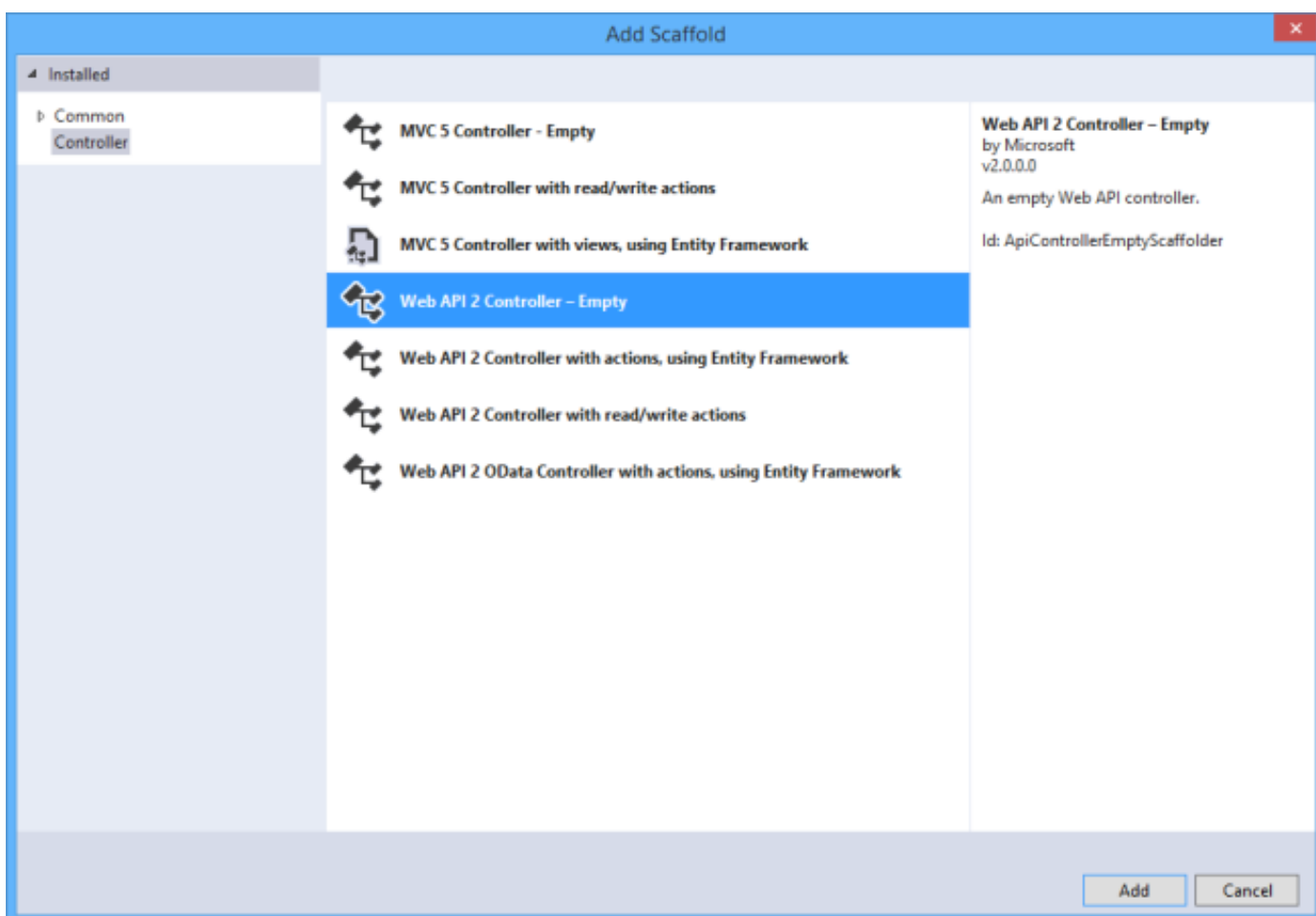
افزودن یک کنترلر

در Web API کنترلرها آبجکت هایی هستند که درخواست های HTTP را مدیریت کرده و آنها را به اکشن متدها نگاشت می کنند. ما کنترلری خواهیم ساخت که می تواند لیستی از محصولات، یا محصولی بخصوص را بر اساس شناسه برگرداند. اگر از ASP.NET MVC استفاده کرده اید، با کنترلرها آشنا هستید. کنترلرهای Web API مشابه کنترلرهای MVC هستند، با این تفاوت که بجای ارث بری از کلاس Controller از کلاس ApiController مشتق می شوند.

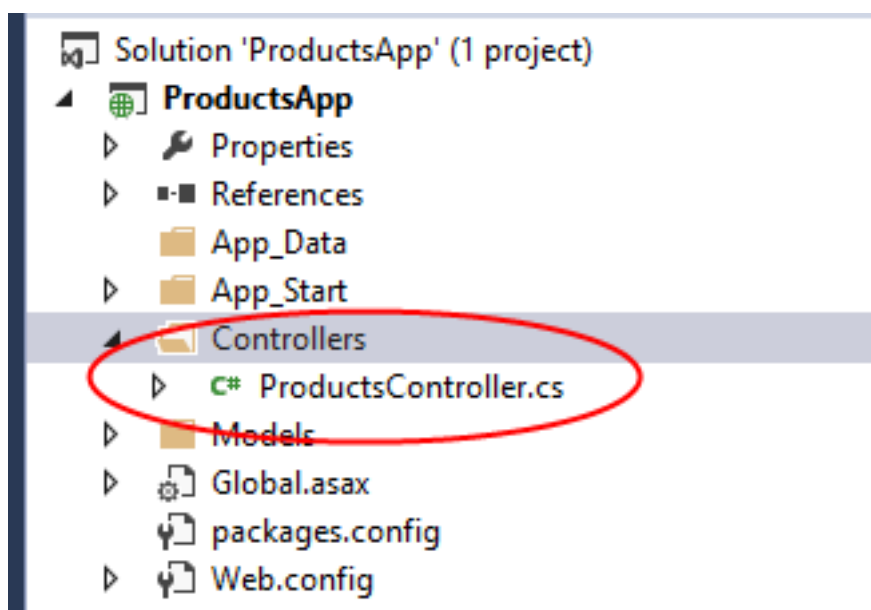
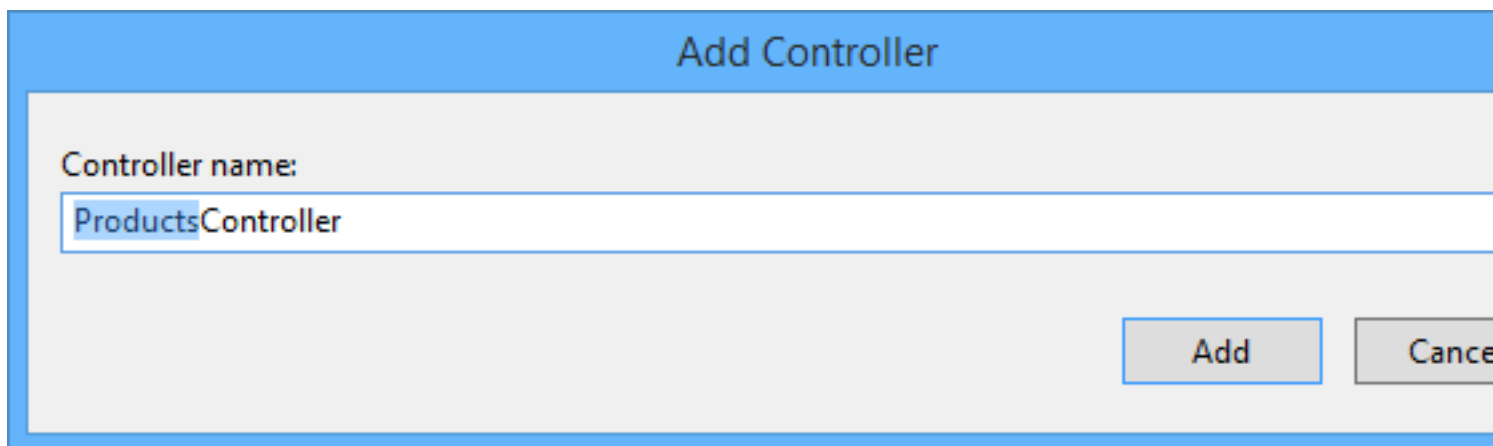
کنترلر جدیدی در پوشه Controllers ایجاد کنید.



در دیالوگ Add Scaffold گزینه Web API Controller - Empty را انتخاب کرده و روی Add کلیک کنید.



در دیالوگ Add Controller نام کنترلر را به "ProductsController" تغییر دهید و روی Add کلیک کنید.



توجه کنید که ملزم به ساختن کنترلرهای خود در پوشه Controllers نیستید، و این روش صرفاً قراردادی برای مرتب نگاه داشتن ساختار پروژه‌ها است. کنترلر ساخته شده را باز کنید و کد زیر را به آن اضافه نمایید.

```
using ProductsApp.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Web.Http;

namespace ProductsApp.Controllers
{
    public class ProductsController : ApiController
    {
        Product[] products = new Product[]
        {
            new Product { Id = 1, Name = "Tomato Soup", Category = "Groceries", Price = 1 },
            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 IHttpActionResult GetProduct(int id)
{
    var product = products.FirstOrDefault((p) => p.Id == id);
    if (product == null)
    {
        return NotFound();
    }
    return Ok(product);
}
}
```

برای اینکه مثال جاری را ساده نگاه داریم، محصولات مورد نظر در یک آرایه استاتیک ذخیره شده اند. مسلماً در یک اپلیکیشن واقعی برای گرفتن لیست محصولات از دیتابیس یا منبع داده ای دیگر کوئری می‌گیرید.

کنترلر ما دو متد برای دریافت محصولات تعریف می‌کند:

متد `GetAllProducts` لیست تمام محصولات را در قالب یک `IEnumerable<Product>` بر می‌گرداند. متد `GetProductById` سعی می‌کند محصولی را بر اساس شناسه تعیین شده پیدا کند.

همین! حالا یک Web API ساده دارید. هر یک از متدهای این کنترلر، به یک یا چند URI پاسخ می‌دهند:

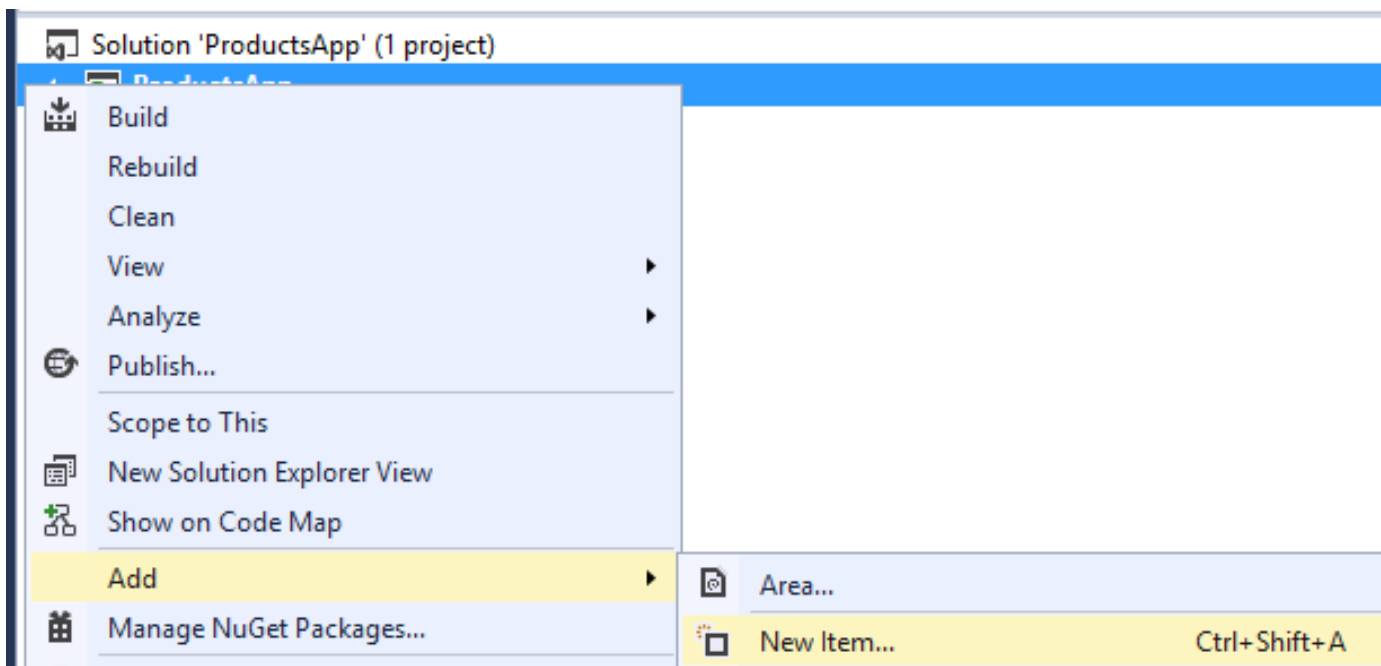
Controller Method	URI
<code>GetAllProducts</code>	<code>api/products/</code>
<code>GetProductById</code>	<code>api/products/ id /</code>

برای اطلاعات بیشتر درباره نحوه نگاشت درخواست‌های HTTP به اکشن متدها توسط Web API به [این لینک](#) مراجعه کنید.

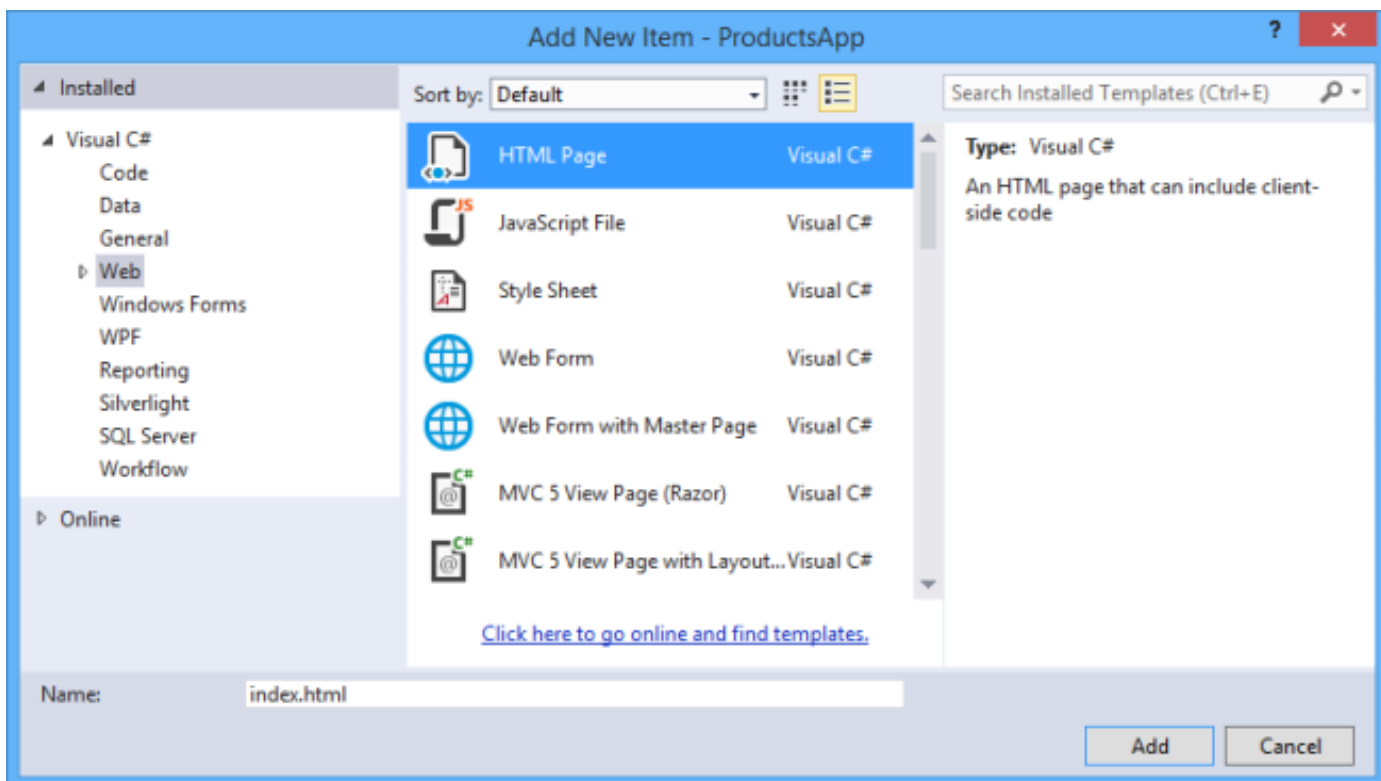
فراخوانی Web API با جاوا اسکریپت و jQuery

در این قسمت یک صفحه HTML خواهیم ساخت که با استفاده از AJAX متدهای Web API را فراخوانی می‌کند. برای ارسال درخواست‌های آژاکسی و بروز رسانی صفحه بمنظور نمایش نتایج دریافتی از jQuery استفاده میکنیم.

در پنجره Solution Explorer روی نام پروژه کلیک راست کرده و گزینه `Add, New Item` را انتخاب کنید.



در دیالوگ Add New Item قالب HTML Page را انتخاب کنید و نام فایل را به "index.html" تغییر دهید.



حال محتوای این فایل را با لیست زیر جایگزین کنید.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
```



```

<title>Product App</title>
</head>
<body>

  <div>
    <h2>All Products</h2>
    <ul id="products" />
  </div>
  <div>
    <h2>Search by ID</h2>
    <input type="text" id="prodId" size="5" />
    <input type="button" value="Search" onclick="find();" />
    <p id="product" />
  </div>

  <script src="http://ajax.aspnetcdn.com/ajax/jquery/jquery-2.0.3.min.js"></script>
  <script>
    var uri = 'api/products';

    $(document).ready(function () {
      // Send an AJAX request
      $.getJSON(uri)
        .done(function (data) {
          // On success, 'data' contains a list of products.
          $.each(data, function (key, item) {
            // Add a list item for the product.
            $('<li>', { text: formatItem(item) }).appendTo($('#products'));
          });
        });

      function formatItem(item) {
        return item.Name + ': $' + item.Price;
      }

      function find() {
        var id = $('#prodId').val();
        $.getJSON(uri + '/' + id)
          .done(function (data) {
            $('#product').text(formatItem(data));
          })
          .fail(function (jqXHR, textStatus, err) {
            $('#product').text('Error: ' + err);
          });
      }
    });
  </script>
</body>
</html>

```

راه‌های مختلفی برای گرفتن jQuery وجود دارد، در این مثال از [Microsoft Ajax CDN](http://ajax.aspnetcdn.com/ajax/jquery/jquery-2.0.3.min.js) استفاده شده. می‌توانید این کتابخانه را از <http://jquery.com> دانلود کنید و بصورت محلی استفاده کنید. همچنین قالب پروژه‌های Web API این کتابخانه را به پروژه نیز اضافه می‌کنند.

گرفتن لیستی از محصولات

برای گرفتن لیستی از محصولات، یک درخواست HTTP GET به آدرس "/api/products" ارسال کنید.

تابع [getJSON](#) یک درخواست آژاکسی ارسال می‌کند. پاسخ دریافتی هم آرایه ای از آبجکت‌های JSON خواهد بود. تابع done در صورت موفقیت آمیز بودن درخواست، اجرا می‌شود. که در این صورت ما DOM را با اطلاعات محصولات بروز رسانی می‌کنیم.

```

$(document).ready(function () {
  // Send an AJAX request
  $.getJSON(apiUrl)
    .done(function (data) {
      // On success, 'data' contains a list of products.
      $.each(data, function (key, item) {
        // Add a list item for the product.
        $('<li>', { text: formatItem(item) }).appendTo($('#products'));
      });
    });
});

```

گرفتن محصولی مشخص

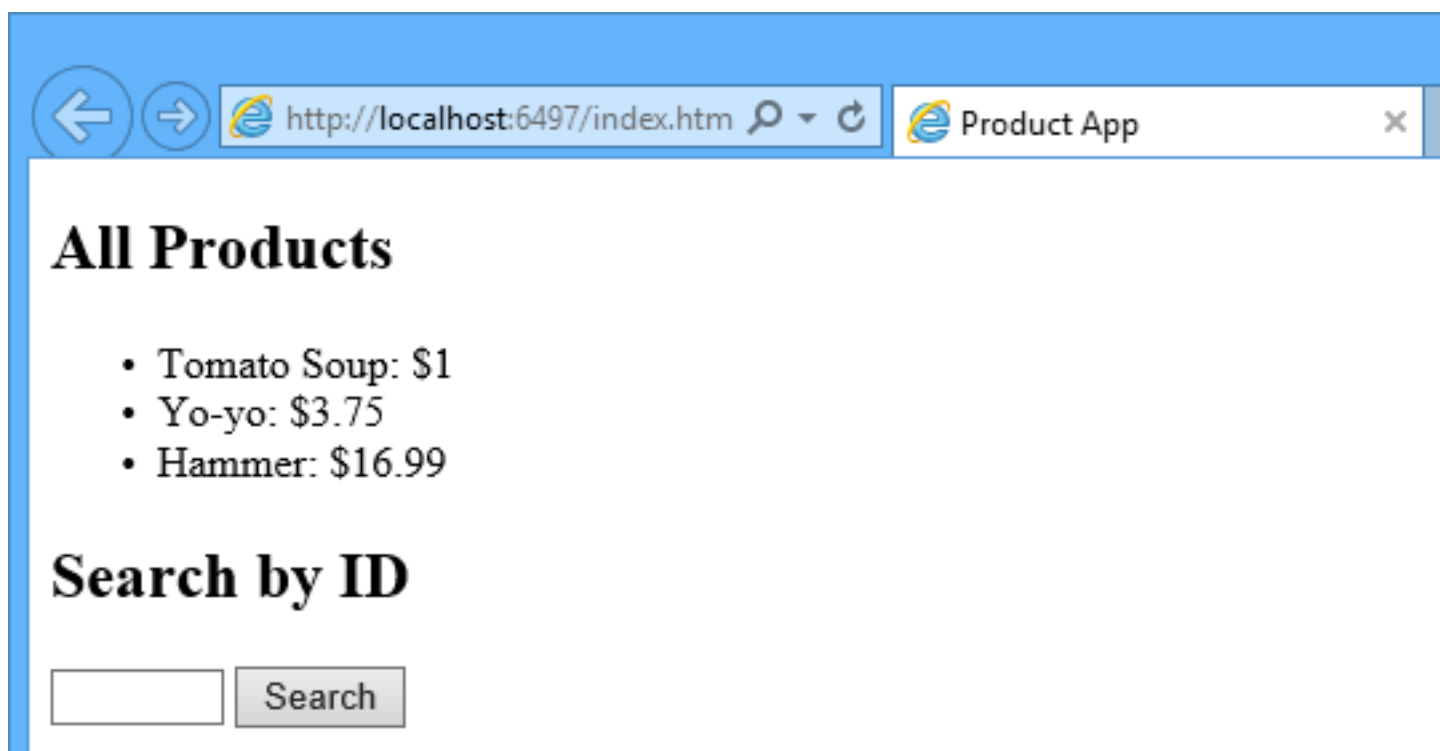
برای گرفتن یک محصول توسط شناسه (ID) آن کافی است یک درخواست HTTP GET به آدرس "/api/products/id" ارسال کنید.

```
function find() {
    var id = $('#prodId').val();
    $.getJSON(apiUrl + '/' + id)
        .done(function (data) {
            $('#product').text(formatItem(data));
        })
        .fail(function (jqXHR, textStatus, err) {
            $('#product').text('Error: ' + err);
        });
}
```

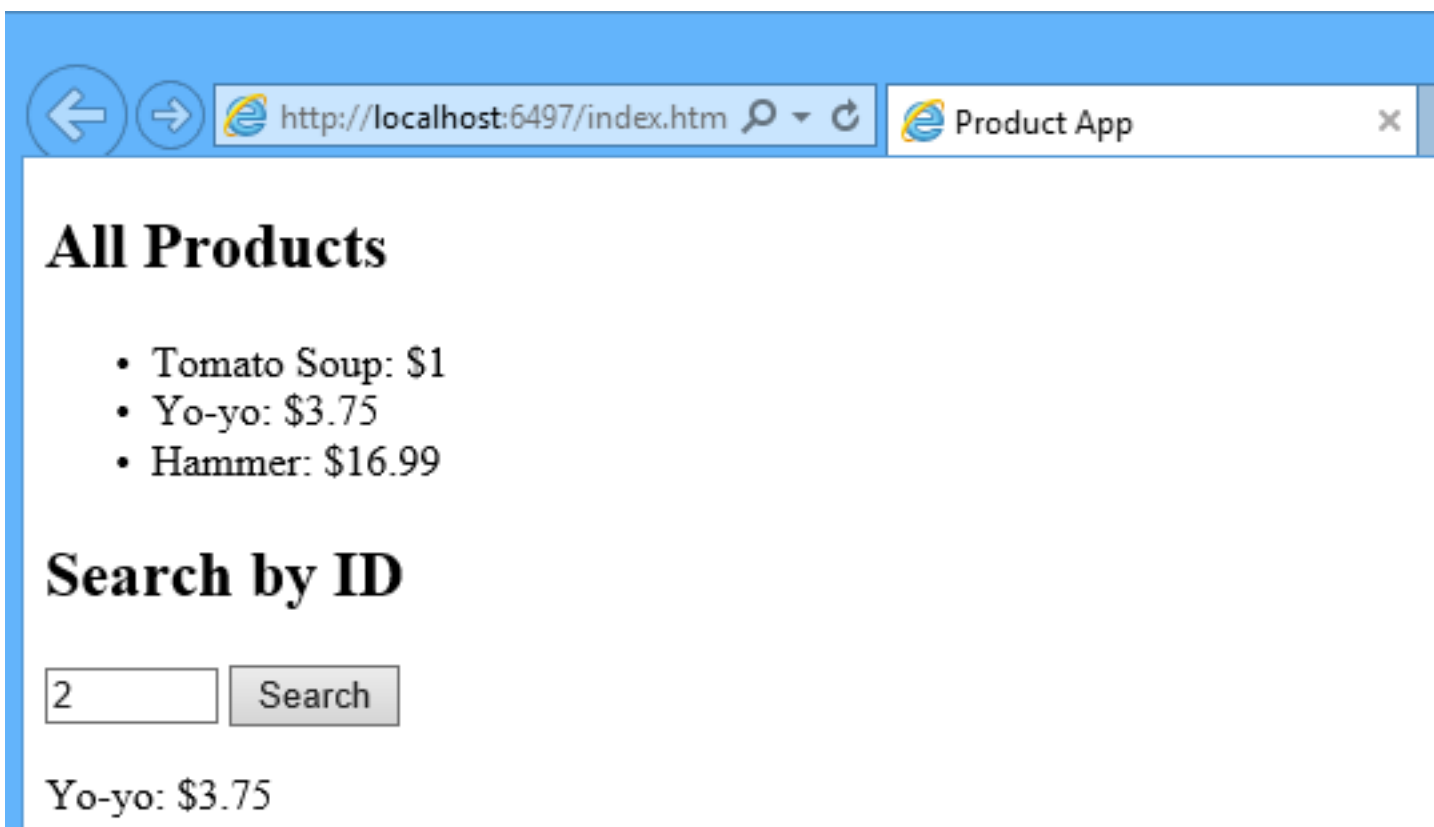
برای این کار هنوز از getJSON برای ارسال درخواست آژاکسی استفاده می‌کنیم، اما اینبار شناسه محصول را هم به آدرس درخواستی اضافه کرده ایم. پاسخ دریافتی از این درخواست، اطلاعات یک محصول با فرمت JSON است.

اجرای اپلیکیشن

اپلیکیشن را با F5 اجرا کنید. صفحه وب باز شده باید چیزی مشابه تصویر زیر باشد.



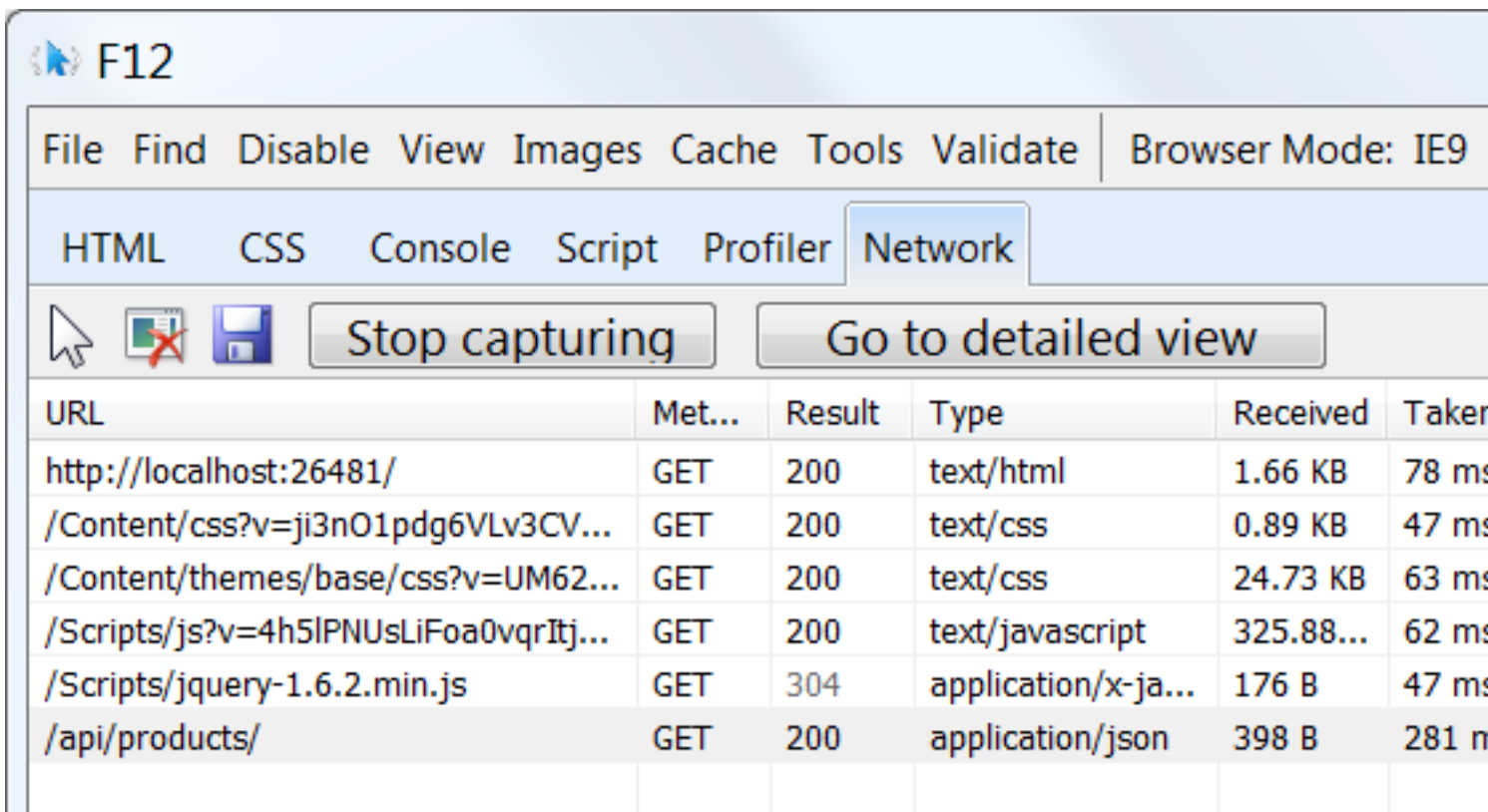
برای گرفتن محصولی مشخص، شناسه آن را وارد کنید و روی Search کلیک کنید.






اگر شناسه نامعتبری وارد کنید، سرور یک خطای HTTP بر می‌گرداند.

استفاده از F12 برای مشاهده درخواست‌ها و پاسخ‌ها

هنگام کار با سرویس‌های HTTP، مشاهده‌ی درخواست‌های ارسال شده و پاسخ‌های دریافتی بسیار مفید است. برای اینکار می‌توانید از ابزار توسعه دهندگان وب استفاده کنید، که اکثر مرورگرهای مدرن، پیاده سازی خودشان را دارند. در اینترنت اکسپلورر می‌توانید با F12 به این ابزار دسترسی پیدا کنید. به برگه Network بروید و روی Start Capturing کلیک کنید. حالا صفحه وب را مجدداً بارگذاری (reload) کنید. در این مرحله اینترنت اکسپلورر ترافیک HTTP بین مرورگر و سرور را تسخیر می‌کند. می‌توانید تمام ترافیک HTTP روی صفحه جاری را مشاهده کنید.



به دنبال آدرس نسبی `/api/products` بگردید و آن را انتخاب کنید. سپس روی `Go to detailed view` کلیک کنید تا جزئیات ترافیک را مشاهده کنید. در نمای جزئیات، می‌توانید headerها و بدنه درخواست‌ها و پاسخ‌ها را ببینید. مثلاً اگر روی برگه `Request headers` کلیک کنید، خواهید دید که اپلیکیشن ما در `Accept header` داده‌ها را با فرمت `"application/json"` درخواست کرده است.

HTML CSS Console Script Profiler Network						
   Stop capturing Back to summary view < Prev						
URL: http://localhost:26481/api/products/						
Request headers		Request body	Response headers	Response body	Cookies	Initiator
Key		Value				
Request		GET /api/products/ HTTP/1.1				
X-Requested-With		XMLHttpRequest				
Accept		application/json, text/javascript, */*; q=0.01				
Referer		http://localhost:26481/				
Accept-Language		en-us				
Accept-Encoding		gzip, deflate				
User-Agent		Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0)				
Host		localhost:26481				
Connection		Keep-Alive				

اگر روی برگه Response body کلیک کنید، می‌توانید ببینید چگونه لیست محصولات با فرمت JSON سریال شده است. همانطور که گفته شده مرورگرهای دیگر هم قابلیت‌های مشابهی دارند. یک ابزار مفید دیگر [Fiddler](#) است. با استفاده از این ابزار می‌توانید تمام ترافیک HTTP خود را مانیتور کرده، و همچنین درخواست‌های جدیدی بسازید که این امر کنترل کاملی روی HTTP headers به شما می‌دهد.

قدم‌های بعدی

برای یک مثال کامل از سرویس‌های HTTP که از عملیات POST, PUT و DELETE پشتیبانی می‌کند به [این لینک](#) مراجعه کنید. برای اطلاعات بیشتر درباره طراحی واکنش گرا در کنار سرویس‌های HTTP به [این لینک](#) مراجعه کنید، که اپلیکیشن‌های تک صفحه ای (SPA) را بررسی می‌کند.

نظرات خوانندگان

نویسنده: پوریا منفرد
تاریخ: ۱۳۹۳/۰۳/۰۵ ۱:۲۶

من به سوالی برام پیش اومده اینه که همیشه از Web API برای پروژههای بزرگ مبتنی بر روی HTTP استفاده کرد؟ منظورم از پروژههای بزرگ یعنی Request هایی که شاید اطلاعات برگشتی مثلا بیش از 1000 رکورد باشه آیا شدنیه؟
یعنی منبع داده بتونه بوسیله Web API عملیاتهای Crud رو بر روی بستر اینترنت برای پروژههای این چنینی که امکان واکنشی اطلاعات بشمار و ورود اطلاعات همزمان بوسیله کاربرهای مختلف با دیوایسهای مختلف وجود داره رو ارائه بده؟

نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۳/۰۳/۰۷ ۰:۷

بله. به طور کلی، هر پلتفرمی که دارای کتابخانه ای جهت کار با سرویسهای Http است میتواند از سرویسهای Asp.Net WebApi استفاده نماید.

اما در هنگام پیاده سازی پروژههای مقیاس بزرگ حتما به طراحی زیر ساخت توجه ویژه ای داشته باشید. اگر کتابهای

[Designing Evolvable Web Api With Asp.Net](#) یا

[Pro Asp.Net Web Api : Http Web Service In Asp.Net](#) را مطالعه نکردید بهتون پیشنهاد میکنم قبل از شروع به کار حتما نگاهی به

آنها بیندازید.

در همین رابطه:

[«مقایسه بین امکانات Web Api و WCF»](#)