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

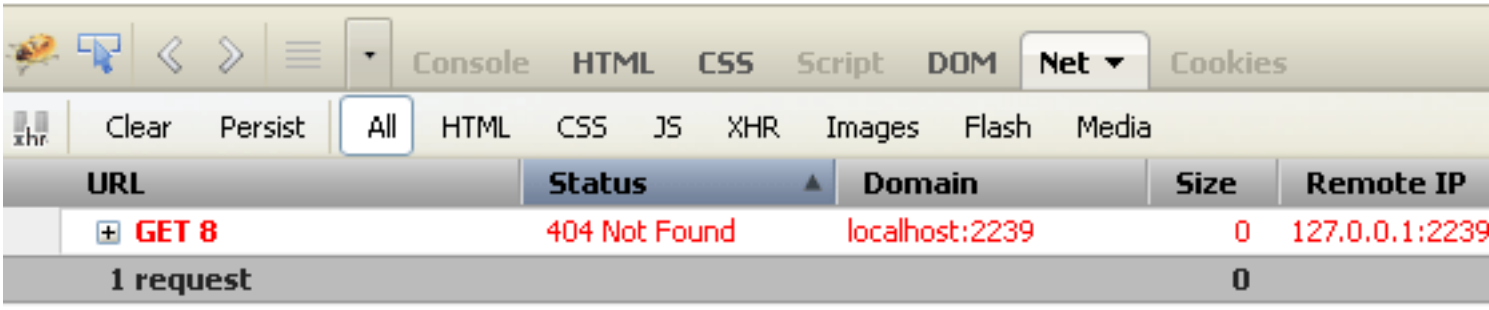
تمامی پاسخ‌های دریافتی از Web API توسط Client، باید در قالب کدهای وضعیت HTTP باشند. دو کلاس جدید با نام‌های HttpResponseMessage و HttpResponseException همراه با ASP.NET MVC 4 معرفی شده اند که ارسال کدهای وضعیت پردازش درخواست به Client را آسان می‌سازند. به عنوان مثال، ارسال وضعیت برای چهار عمل اصلی بازیابی،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

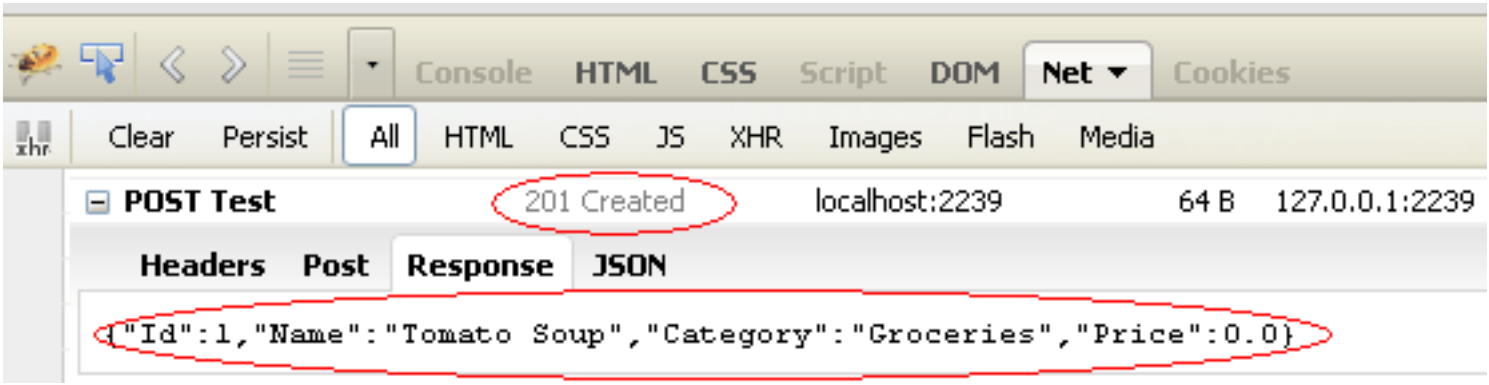
این پیاده سازی کار می‌کند اما کمبودهایی دارد:

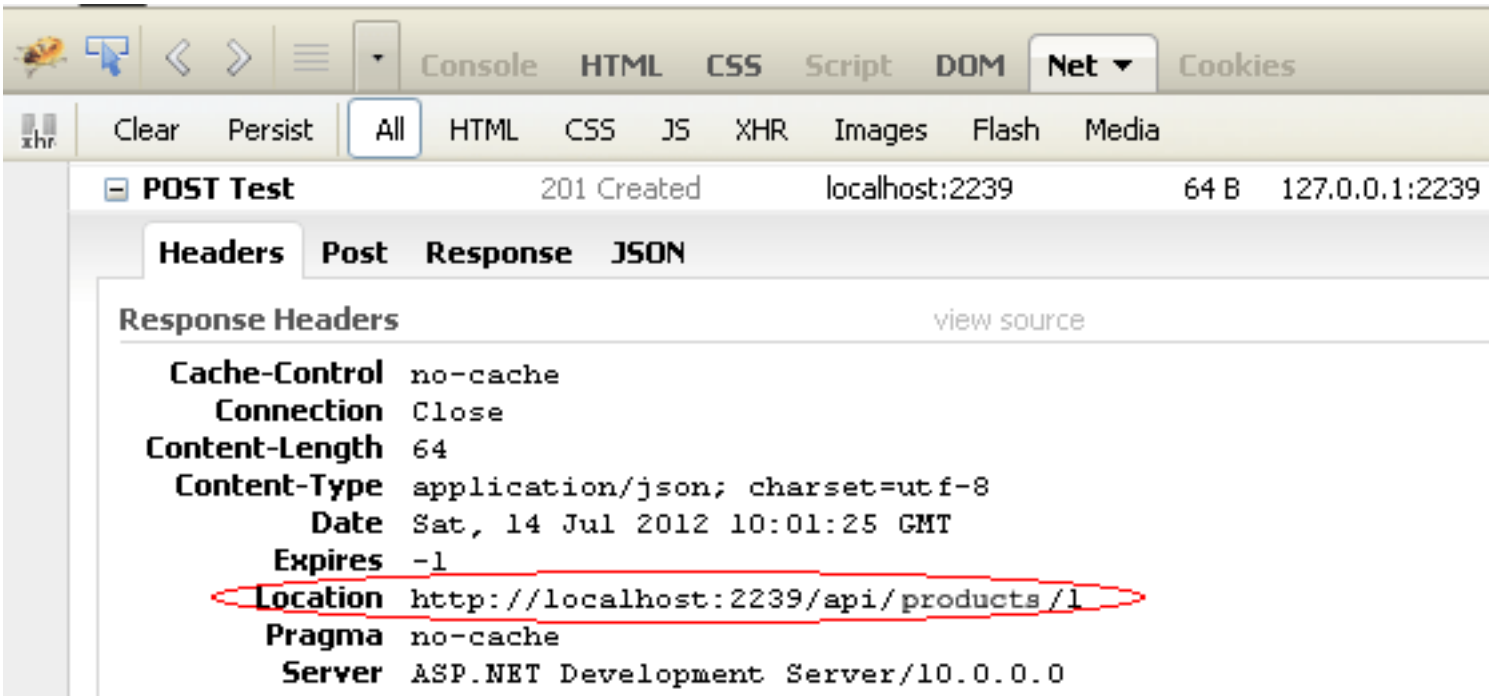
کد وضعیت پردازش درخواست : به طور پیش فرض، Web API، کد 200 را در پاسخ ارسال می‌کند، اما بر اساس مستندات پروتوکل HTTP، زمانی که یک درخواست از نوع POST منجر به تولید منبعی می-شود، Server باید کد وضعیت 201 را به Client برگشت بدهد.
آدرس منبع جدید ایجاد شده : بر اساس مستندات پروتوکل HTTP، زمانی که منبعی بر 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 int، string 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.

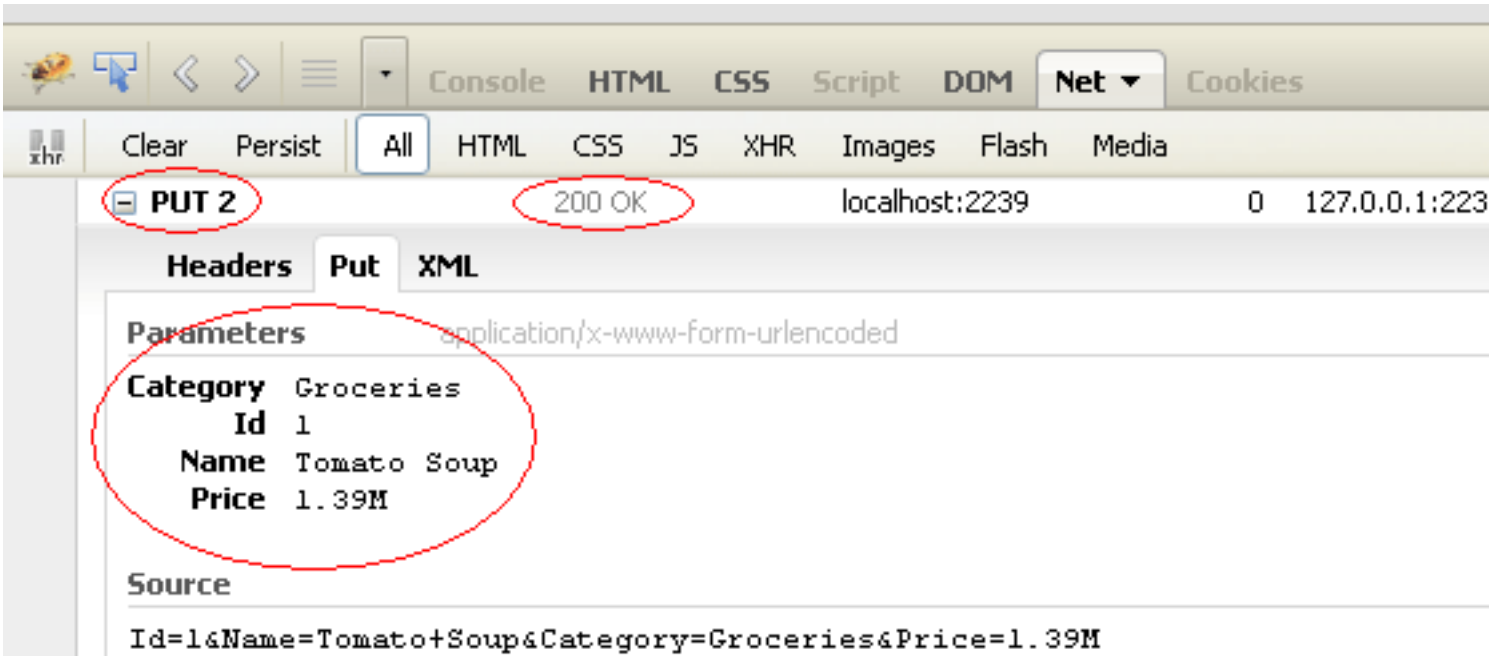
فراخوانی متدها و مدیریت کدهای وضعیت HTTP در سمت Client

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.

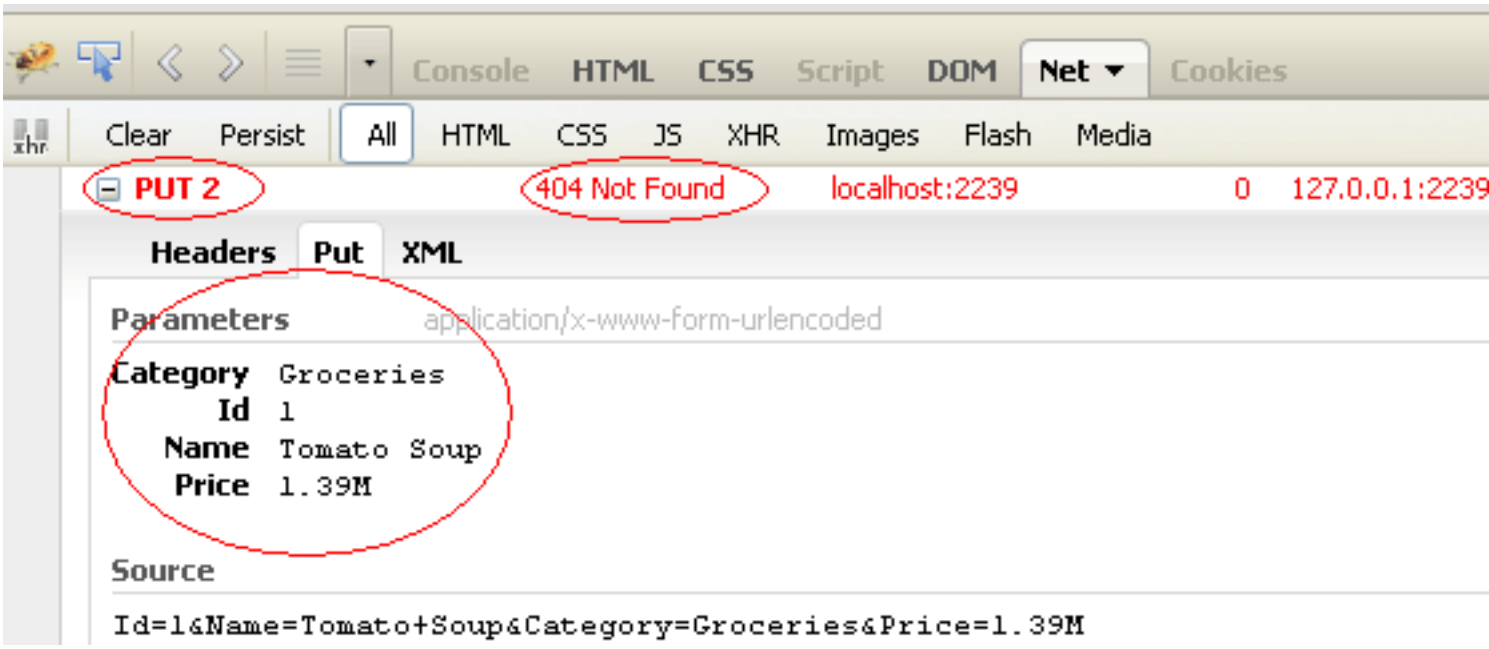
```
var id = $("#myTextBox").val();

$.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.