عنوان: Best Practice هایی برای طراحی RESTful API - قسمت اول
نویسنده: محسن درپرستی
تاریخ: ۱۳۹۲/۱۰/۱۶ ۱۷:۰
آدرس: www.dotnettips.info
گروه‌ها: ASP.NET Web API, RESTful API, Best Practice

با آمدن Asp.Net Web API کار ساختن Web API ها برای برنامه نویس‌ها به خصوص دسته ای که با ساخت API و وب سرویس آشنا نبودند خیلی ساده‌تر شد . اگر با Asp.Net MVC آشنا باشید خیلی سریع می‌توانید اولین Web Service خودتان را بسازید .

در صفحه مربوط به Asp.Net Web API آمده است که این فریمورک بستر مناسبی برای ساخت و توسعه برنامه های RESTful است . اما تنها ساختن کنترلر و اکشن و برگشت دادن داده‌ها به سمت کلاینت ، به خودی خود برنامه شما رو تبدیل به یک RESTful API نمی‌کند .

مثل تمام مفاهیم و ابزارها ، طراحی و ساختن RESTful API هم دارای اصول و Best Practice هایی است که رعایت آنها به خصوص در این زمینه از اهمیت زیادی برخوردار است . همانطور که از تعریف API برمی آید شما در حال طراحی رابطی هستید تا به توسعه دهندگان دیگر امکان دهید از داده‌ها و یا خدمات شما در برنامه‌ها و سرویس هایشان استفاده کنند . مانند APIهای توئیتر و نقشه گوگل که برنامه‌های زیادی بر مبنای آنها ساخته شده اند . در واقع توسعه دهندگان مشتریان API شما هستند .

بهره وری توسعه دهنده مهمترین اصل

اینطور می‌توان نتیجه گرفت که اولین و مهمترین اصل در طراحی API باید رضایت و موفقیت توسعه دهنده در درک و یادگیری سریع API شما ،نه تنها با کمترین زحمت بلکه همراه با حس نشاط ، باشد. (تجربه کاربری در اینجا هم می‌تواند صدق کند). سعی کنید در زمان انتخاب از بین روش‌های طراحی موجود ، از دیدگاه توسعه دهنده به مسئله نگاه کنید . خود را به جای او قرار دهید و تصور کنید که می‌خواهید با استفاده از API موجود یک رابط کاربری طراحی کنید یا یک اپلیکشن برای موبایل بنویسید و اصل را این نکته قرار دهید که بهره وری برنامه نویس را حداکثر کنید. ممکن است گاهی بین طرحی که بر اساس این اصل برای API خود در نظر داریم و یکی از اصول یا استانداردها تعارض بوجود بیاید . در این موارد بعد از اینکه مطمئن شدیم این اختلاف ناشی از طراحی و درک اشتباه خودمان نیست (که اکثرا هست) ارجحیت را باید به طراحی بدهیم .

تهیه مستندات API

اگر برای پروژه وب سایتتان هیچ نوشته ای یا توضیحی ندارید ، جالب نیست اما خودتان ساختار برنامه خود را می‌شناسید و کار را پیش می‌برید. اما توسعه دهنده ای که از API شما می‌خواهد استفاده کند و به احتمال زیاد شما را نمی‌شناسد ، عضو تیم شما هم نیست ، هیچ ایده ای درباره ساختار آن ، روش نامگذاری توابع و منابع، ساختار Urlها ، چگونگی و گام‌های پروسه درخواست تا دریافت پاسخ ندارد ،و به مستندات شما وابسته است و تمام اینها باید در مستندات شما باشد. بیشتر توسعه دهنده‌ها قبل از تست کردن API شما سری به مستندات می‌زنند ، دنبال نمونه کد آموزشی می‌گردند و در اینترنت درباره آن جستجو می‌کنند .
ازینرو مستندات (کارامد) یک ضرورت است :
1- در مستندات باید هم درباره کلیت و هم در مورد تک تک توابع (پارامترهای معتبر ، ساختار پاسخ‌ها و ...) توضیحات وجود داشته باشد.
2- باید شامل مثالهایی از سیکل کامل درخواست‌ها / پاسخ‌ها باشند .
3- تغییرات اعمال شده نسبت به نسخه‌های قبلی باید در مستندات بیان شوند .
4- (در وب) یافتن و جستجو کردن در مستنداتی که به صورت فایل Pdf هستند یا برای دسترسی نیاز به Login داشته باشند سخت و آزاردهنده هستند.
5- کسی را داشته باشید تا با و بدون مستندات با API شما کار کند و از این روش برای تکمیل و اصلاح مستندات استفاده کنید.

رعایت نسخه بندی و حفظ نسخه‌های قبلی به صورت فعال برای مدت معین

یک API تقریبا هیچوقت کاملا پایدار نمی‌شود و اعمال تغییرات برای بهبود آن اجتناب ناپذیر هستند . مسئله مهم این است که چطور این تغییرات مدیریت شوند . مستند کردن تغییرات ، اعلام به موقع آنها و دادن یک بازه زمانی کافی برای ارتقا یافتن برنامه

هایی که از نسخه‌های قدیمی‌تر استفاده می‌کنند نکات مهمی هستند . همیشه در کنار نسخه بروز و اصلی یک یا دو نسخه (بسته به API و کلاینت‌های آن) قدیمی‌تر را برای زمان مشخصی در حالت سرویس دهی داشته باشید .

داشتن یک روش مناسب برای اعلام تغییرات و ارائه مستندات و البته دریافت بازخورد از استفاده کنندگان

تعامل با کاربران برنامه باید از کانال‌های مختلف وجود داشته باشد .از وبلاگ ، Google Groups ، Mailing List و دیگر ابزارهایی که در اینترنت وجود دارند برای انتشار مستندات ، اعلام بروزرسانی‌ها ، قرار دادن مقالات و نمونه کدهای آموزشی ، پرسش و پاسخ با کاربران استفاده کنید .

مدیریت خطاها به شکل صحیح که به توسعه دهنده در آزمودن برنامه اش کمک کند.

از منظر برنامه نویسی که از API شما استفاده می‌کند هرآنچه در آنسوی API اتفاق می‌افتد یک جعبه سیاه است . به همین جهت خطاهای API شما ابزار کلیدی برای او هستند که خطایابی و اصلاح برنامه در حال توسعه اش را ممکن می‌کنند . علاوه بر این ، زمانی که برنامه نوشته شده با API شما مورد استفاده کاربر نهایی قرار گرفت ، خطاهای به دقت طراحی شده API شما کمک بزرگی برای توسعه دهنده در عیب یابی هستند .

-1 از Status Code های HTTP استفاده کنید و سعی کنید تا حد ممکن آنها را نزدیک به مفهوم استانداردشان بکار ببرید .

-2 خطا و علت آن را به زبان روشن توضیح دهید و در توضیح خساست به خرج ندهید .

-3 در صورت امکان لینکی به یک صفحه وب که حاوی توضیحات بیشتری است را در خطا بگنجانید .

رعایت ثبات و یکدستی در تمام بخش‌های طراحی که توانایی پیش بینی توسعه دهنده را در استفاده از API افزایش می‌دهد .

داشتن مستندات لازم است اما این بدین معنی نیست که خود API نباید خوانا و قابل پیش بینی باشد . از هر روش و تکنیکی که استفاده می‌کنید آن را در تمام پروژه حفظ کنید . نامگذاری توابع/منابع ، ساختار پاسخ‌ها ، Urlها ، نقش و عملیاتی که HTTP methodها در API شما انجام می‌دهند باید ثبات داشته باشند . از این طریق توسعه دهنده لازم نیست برای هر بخشی از API شما به سراغ فایل‌ها راهنما برود . و به سرعت کار خود را به پیش می‌برد .

انعطاف پذیر بودن API

API توسط کلاینت‌های مختلفی و برای افراد مختلفی مورد استفاده قرار می‌گیرد که لزوما همه‌ی آنها ساختار یکسانی ندارند و API شما باید تا جای ممکن بتواند همه آنها را پوشش دهد . محدود بودن فرمت پاسخ ، ثابت بودن فیلدهای ارسالی به کلاینت ، ندادن امکان صفحه بندی ، مرتب سازی و جستجو در داده‌ها به کلاینت ، داشتن تنها یک نوع احراز هویت ، وابسته بودن به کوکی و ... از مشخصات یک API منجمد و انعطاف ناپذیر هستند .

اینها اصولی کلی بودند که بسیاری از آنها مختص طراحی API نیستند و در تمام حوزه‌ها قابل استفاده بوده ، جز الزامات هستند . در قسمت‌های بعدی نکات اختصاصی‌تری را بررسی خواهیم کرد .

نظرات خوانندگان

نویسنده: وحید
تاریخ: ۱۳۹۲/۱۰/۱۷ ۱:۱۴

با سلام یعنی شما میگویید برای web api میبابیست یک لایه جدا تعریف نمود و cors را در آن لحاظ نمود؟

نویسنده: محسن درپرستی
تاریخ: ۱۳۹۲/۱۰/۱۷ ۹:۱۲

لایه جدا از چه چیزی ؟ اگر منظورتان در Asp.Net باشد ، پروژه هایی که با استفاده از Asp.Net Web API ساخته می‌شوند خود یک سیستم مستقل هستند .

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۰/۲۹ ۲۲:۳۹

سایت silverreader که از asp.net web api استفاده می‌کنه، نگارش اول کارش با این آدرس شروع میشه

https://silverreader.com/api/v1

نویسنده: شهروز جعفری
تاریخ: ۱۳۹۲/۱۲/۰۶ ۷:۳۶

بحث cors یک چیز است و بحث Best Practice هایی برای طراحی RESTful API یک چیز .در کل زمانی که ما می‌خواهیم یک سرویس ارائه دهیم و باقی از آن استفاده کنند داستانش با یک متد که فقط خودمان از آن استفاده می‌کنیم فرق می‌کند. باید به خیلی چیزها حواسمان باشد.

نویسنده: شهروز جعفری
تاریخ: ۱۳۹۲/۱۲/۰۶ ۷:۳۹

من یک سری مطلب درباره نسخه بندی در API پیدا کردم باهاتون به اشتراک میذارم شاید مفید باشه:

http://www.lexicalscope.com/blog/2012/03/12/how-are-rest-apis-versioned/

http://stackoverflow.com/questions/10742594/versioning-rest-api

عنوان: Best Practice هایی برای طراحی RESTful API - قسمت دوم
نویسنده: محسن درپرستی
تاریخ: ۱۳۹۲/۱۲/۰۳ ۱۴:۰
آدرس: www.dotnettips.info
گروه‌ها: ASP.NET Web API, Naming, Web API, RESTful API, Best Practice

طراحی Url در Restful API

Url بخش اصلی و راه ارتباطی API شما با توسعه دهنده است .بنابراین طراحی یک ساختار مناسب و یکپارچه برای Url ها دارای اهمیت زیادی است .

Url پایه API خود را ساده و خوانا ، حفظ کنید . داشتن یک Url پایه ساده استفاده از API را آسان کرده و خوانایی آن را بالا میبرد و باعث می‌شود که توسعه دهنده برای استفاده از آن نیاز کمتری به مراجعه به مستندات داشته باشد. پیشنهاد می‌شود که برای هر منبع تنها دو Url پایه وجود داشته باشد . یکی برای مجموعه ای از منبع موردنظر و دیگری برای یک واحد مشخص از آن منبع .
برای مثال اگر منبع موردنظر ما کتاب باشد ، خواهیم داشت :

```
.../books
```

برای مجموعه‌ی کتابها و

```
.../books/1001
```

برای کتابی با شناسه 1001
استفاده از این روش یک مزیت دیگر هم به همراه دارد و آن دور کردن افعال از Url ها است.

بسیاری در زمان طراحی Url ها و در نامگذاری از فعل‌ها استفاده می‌کنند. برای هر منبعی که مدلسازی می‌کنید هیچ وقت نمی‌توانید آن را به تنهایی و جداافتاده در نظر بگیرید. بلکه همیشه منابع مرتبطی وجود دارند که باید در نظر گرفته شوند. در مثال کتاب می‌توان منابعی مثل نویسنده ، ناشر ، موضوع و ... را بیان کرد. حالا سعی کنید به تمام Url هایی که برای پوشش دادن تمام درخواست‌های مربوط به منبع کتاب نیاز داریم فکر کنید . احتمالا به چیزی شبیه این می‌رسیم :

```
.../getAllBooks
.../getBook
.../newBook
.../getNewBooksSince
.../getComputerBooks
.../BooksNotPublished
.../UpdateBookPriceTo
.../bookForPublisher
.../GetLastBooks
.../DeleteBook
….
```

خیلی زود یک لیست طولانی از Url ها خواهید داشت که به علت نداشتن یک الگوی ثابت و مشخص استفاده از API شما را واقعا سخت می‌کند.

پس حالا این درخواست‌های متنوع را چطور با دو Url اصلی انجام دهیم ؟
-1 از افعال Http برای کار کردن بر روی منابع استفاده کنید . با استفاده از افعال Http شامل PUT ، GET ، POST و DELETE و دو Url اصلی ، یک مجموعه‌ی مناسب از عملیات‌ها در دسترس توسعه دهنده خواهد بود . به جدول زیر نگاه کنید .

منبع	POST Create	GET Read	PUT Update	DELETE Delete
/books	ثبت کتاب جدید	لیست کتابها	بروزرسانی کلی کتابها	حذف تمام کتابها
/books/1001	خطا	نمایش کتاب ۱۰۰۱	اگر وجود داشته باشد بروزرسانی وگرنه خطا	حذف کتاب ۱۰۰۱

توسعه دهندگان احتمالا نیازی به این جدول برای درک اینکه API چطور کار می‌کند نخواهند داشت.

-2 با استفاده از نکته قبلی بخشی از Url های بالا حذف خواهند شد. اما هنوز با روابط بین منابع چکار کنیم؟ منابع تقریبا همیشه دارای روابطی با دیگر منابع هستند . یک روش ساده برای بیان این روابط در API چیست ؟ به مثال کتاب برمیگردیم. کتاب‌ها دارای نویسنده هستند. اگر بخواهیم کتاب‌های یک نویسنده را برگردانیم چه باید بکنیم؟ با استفاده از Url های پایه و افعال Http می‌توان اینکار را انجام داد. یکی از ساختارهای ممکن این است :

```
GET .../authors/1001/books
```

اگر بخواهیم یک کتاب جدید به کتابهای این نویسنده اضافه کنیم :

```
POST .../authors/1001/books
```

و حدس زدن اینکه برای حذف کتابهای این نویسنده چه باید کرد ، سخت نیست .

-3 بیشتر API ها دارای پیچیدگی‌های بیشتری نسبت به Url اصلی یک منبع هستند . هر منبع مشخصات و روابط متنوعی دارد که قابل جستجو کردن، مرتب سازی، بروزرسانی و تغییر هستند. Url اصلی را ساده نگه دارید و این پیچیدگی‌ها را به کوئری استرینگ منتقل کنید.

برای برگرداندن تمام کتابهای با قیمت پنج هزار تومان با قطع جیبی که دارای امتیاز 8 به بالا هستند از کوئری زیر می‌شود استفاده کرد :

```
GET .../books?price=5000&size=pocket&score=8
```

و البته فراموش نکنید که لیستی از فیلدهای مجاز را در مستندات خود ارائه کنید.

4 - گفتیم که بهتر است افعال را از Url ها خارج کنیم . ولی در مواردی که درخواست ارسال شده در مورد یک منبع نیست چطور؟ مواردی مثل محاسبه مالیات پرداختی یا هزینه بیمه ، جستجو در کل منابع ، ترجمه یک عبارت یا تبدیل واحدها . هیچکدام از اینها ارتباطی با یک منبع خاص ندارند. در این موارد بهتر است از افعال استفاده شود. و حتما در مستندات خود ذکر کنید که در این موارد از افعال استفاده می‌شود.

```
.../convert?value=25&from=px&to=em
.../translate?term=web&from=en&to=fa
```

5 - استفاده از اسامی جمع یا مفرد

با توجه به ساختاری که تا اینجا طراحی کرده ایم بکاربردن اسامی جمع بامعناتر و خواناتر است. اما مهمتر از روشی که بکار می‌برید ، اجتناب از بکاربردن هر دو روش با هم است ، اینکه در مورد یک منبع از اسم منفرد و در مورد دیگری از اسم جمع استفاده کنید . یکدستی API را حفظ کنید و به توسعه دهنده کمک کنید راحت‌تر API شما را یاد بگیرد.

-6 استفاده از نام‌های عینی به جای نام‌های کلی و انتزاعی

API ی را در نظر بگیرید که محتواهایی را در فرمت‌های مختلف ارائه می‌دهد. بلاگ ، ویدئو ، اخبار و .... حالا فرض کنیداین API منابع را در بالاتری سطح مدسازی کرده باشد مثل /items یا /assets . درک کردن محتوای این API و کاری که می‌توان با این API انجام داد برای توسعه دهنده سخت است . خیلی راحت‌تر و مفیدتر است که منابع را در قالب بلاگ ، اخبار ، ویدئو مدلسازی کنیم .