عنوان: **کار با RavenDB از طریق REST API آن**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۶/۱۴ ۱۴:۱۲
آدرس: www.dotnettips.info
برچسب‌ها: NoSQL, RavenDB

در این قسمت قصد داریم برخلاف رویه معمول کار با RavenDB که از طریق کتابخانه‌های کلاینت آن انجام می‌شود، با استفاده از REST API آن، ساز و کار درونی آن‌را بیشتر بررسی کنیم.

REST چیست؟

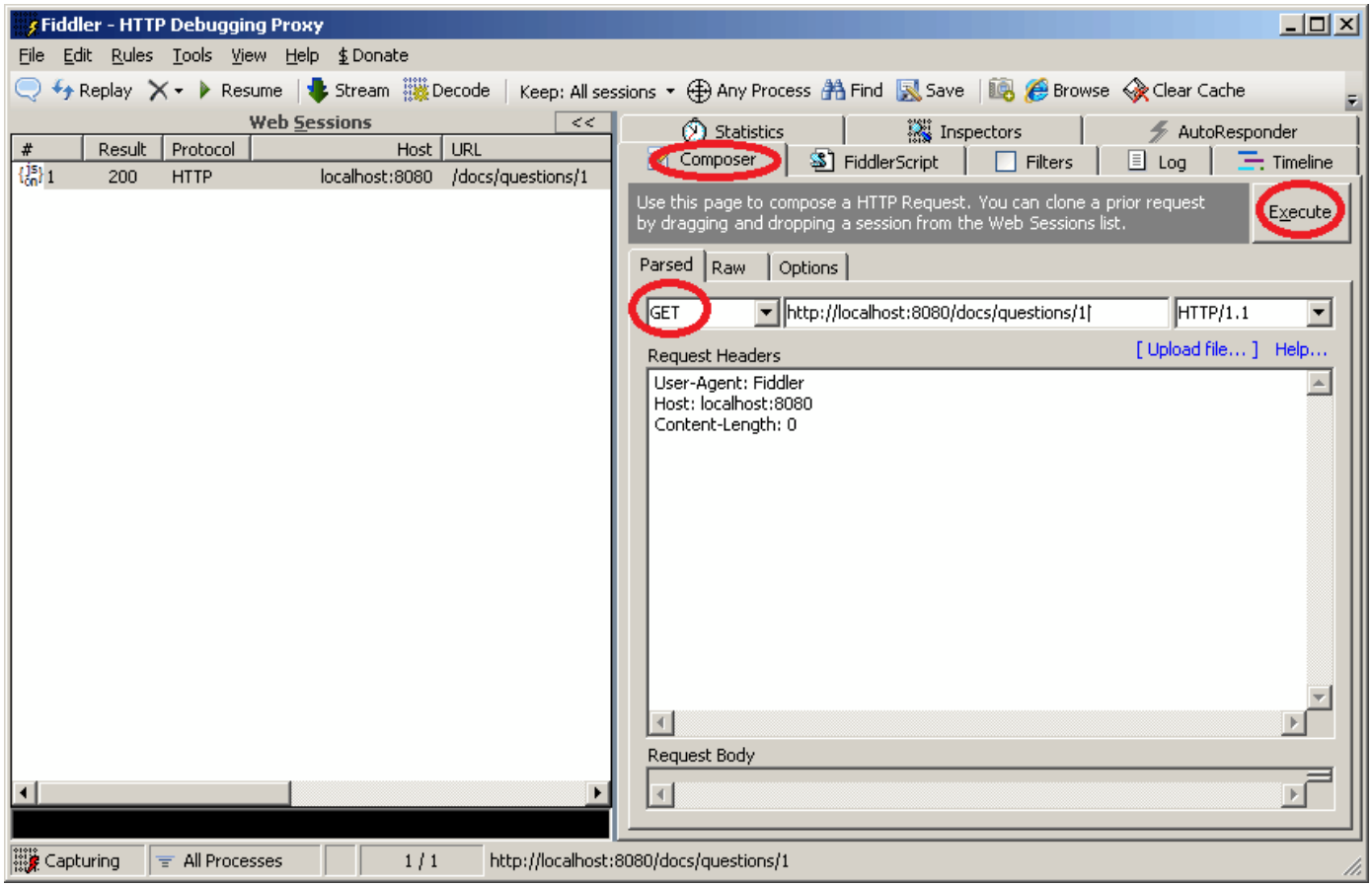
برای درک ساختار پشت صحنه RavenDB نیاز است با مفهوم REST آشنا باشیم؛ زیرا سرور این بانک اطلاعاتی، خود را به صورت یک RESTful web service در اختیار مصرف کنندگان قرار می‌دهد.
REST مخفف representational state transfer است و این روزها هر زمانیکه صحبت از آن به میان می‌آید منظور یک RESTful web service است که با استفاده از تعدادی HTTP Verb استاندارد می‌توان با آن کار کرد؛ مانند GET، POST، PUT و DELETE. با استفاده از GET، یک منبع ذخیره شده بازگشت داده می‌شود. با استفاده از فعل PUT، اطلاعاتی به منابع موجود اضافه و یا جایگزین می‌شوند. POST نیز مانند PUT است با این تفاوت که نوع اطلاعات ارسالی آن اهمیتی نداشته و تفسیر آن به سرور واگذار می‌شود. از DELETE نیز برای حذف یک منبع استفاده می‌گردد.

چند مثال

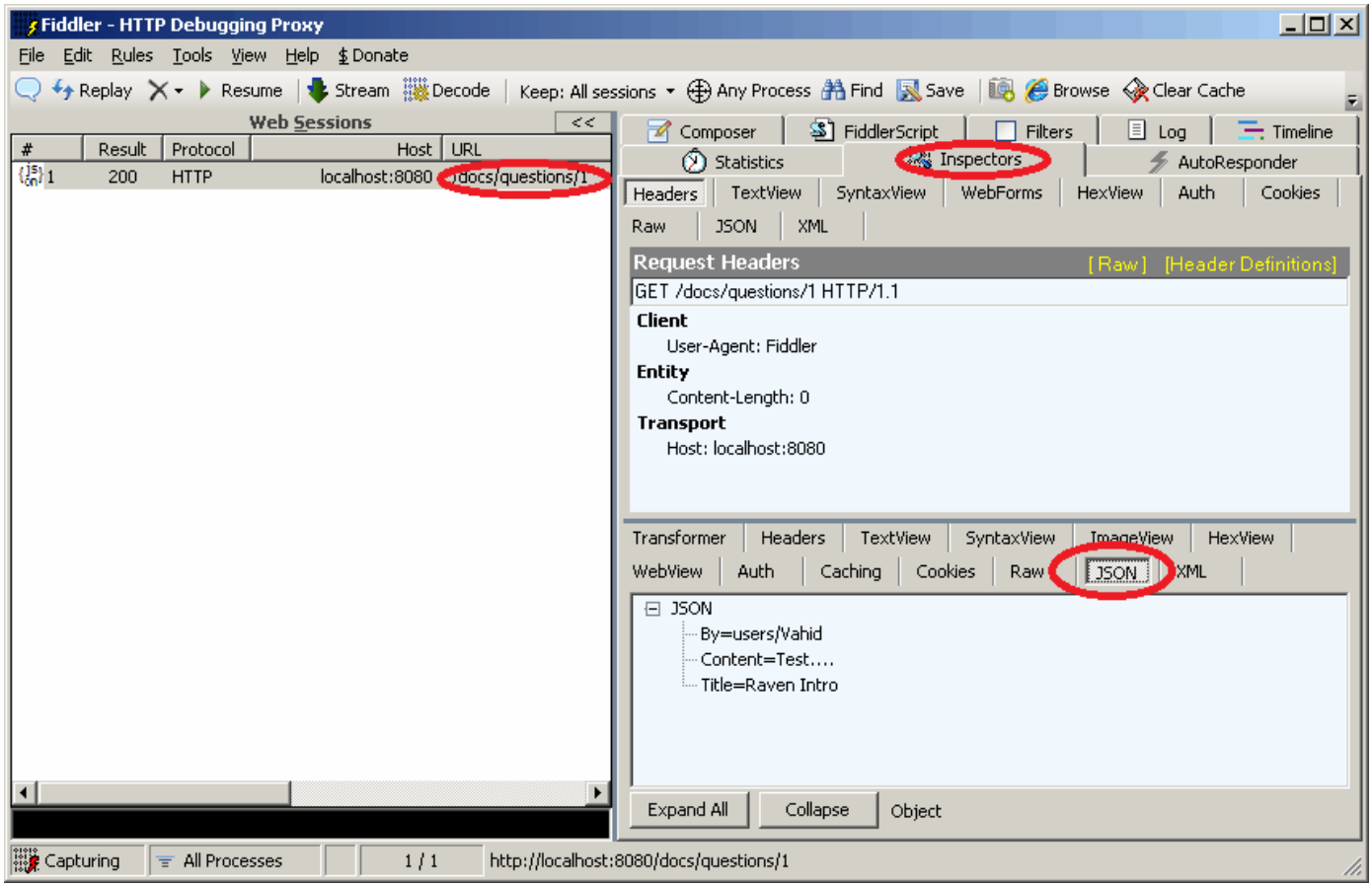
فرض کنید REST API برنامه‌ای از طریق آدرس http://myapp.com/api/questions در اختیار شما قرار گرفته است. در این آدرس، به questions منابع یا Resource گفته می‌شود. اگر دستور GET پروتکل HTTP بر روی این آدرس اجرا شود، انتظار ما این است که لیست تمام سؤالات بازگشت داده شود و اگر از دستور POST استفاده شود، باید یک سؤال جدید به مجموعه منابع موجود اضافه گردد.
اکنون آدرس http://myapp.com/api/questions/1 را درنظر بگیرید. در اینجا عدد یک معادل Id اولین سؤال ثبت شده است. بر اساس این آدرس خاص، اینبار اگر دستور GET صادر شود، تنها اطلاعات سؤال یک بازگشت داده خواهد شد و یا اگر از دستور PUT استفاده شود، اطلاعات سؤال یک با مقدار جدید ارسالی جایگزین می‌شود و یا با فراخوانی دستور DELETE، سؤال شماره یک حذف خواهد گردید.

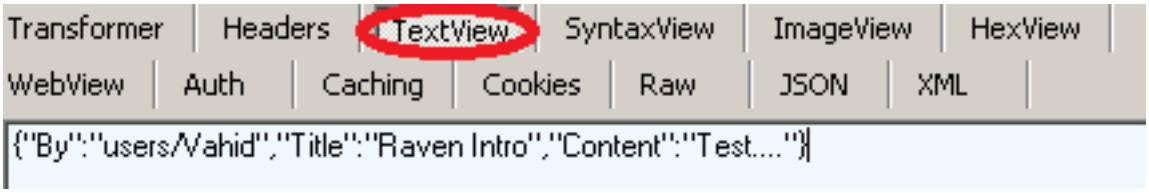
کار با دستور GET

در ادامه، به مثال قسمت قبل مراجعه کرده و تنها سرور RavenDB را اجرا نمائید (برنامه Raven.Server.exe)، تا در ادامه بتوانیم دستورات HTTP را بر روی آن امتحان کنیم. همچنین نیاز به برنامه معروف فیدلر نیز خواهیم داشت. از این برنامه برای ساخت دستورات HTTP استفاده خواهد شد.
پس از دریافت و نصب فیدلر، برگه Composer آن‌را گشوده و http://localhost:8080/docs/questions/1 را در حالت GET اجرا کنید:

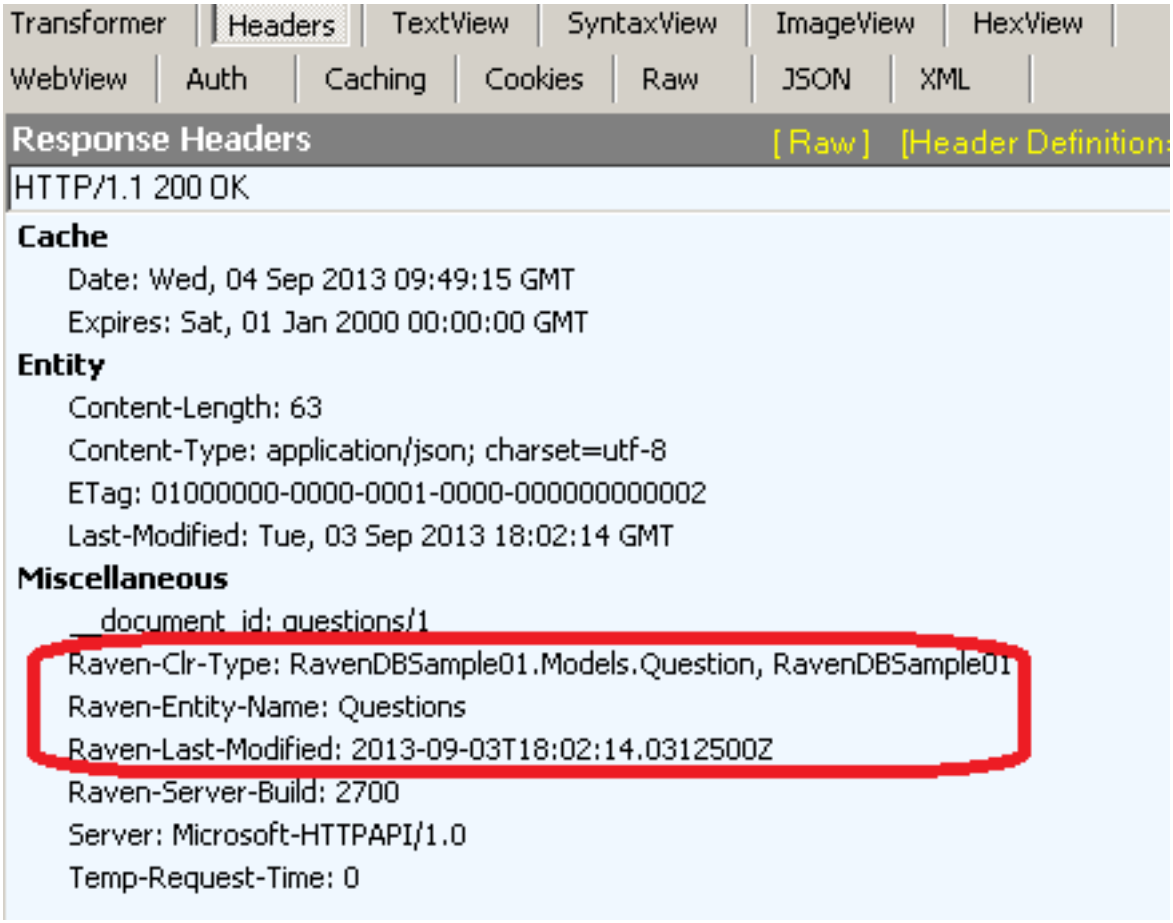


در این حالت دستور بر روی بانک اطلاعاتی اجرا شده و خروجی را در برگه Inspectors آن می‌توان مشاهده کرد:



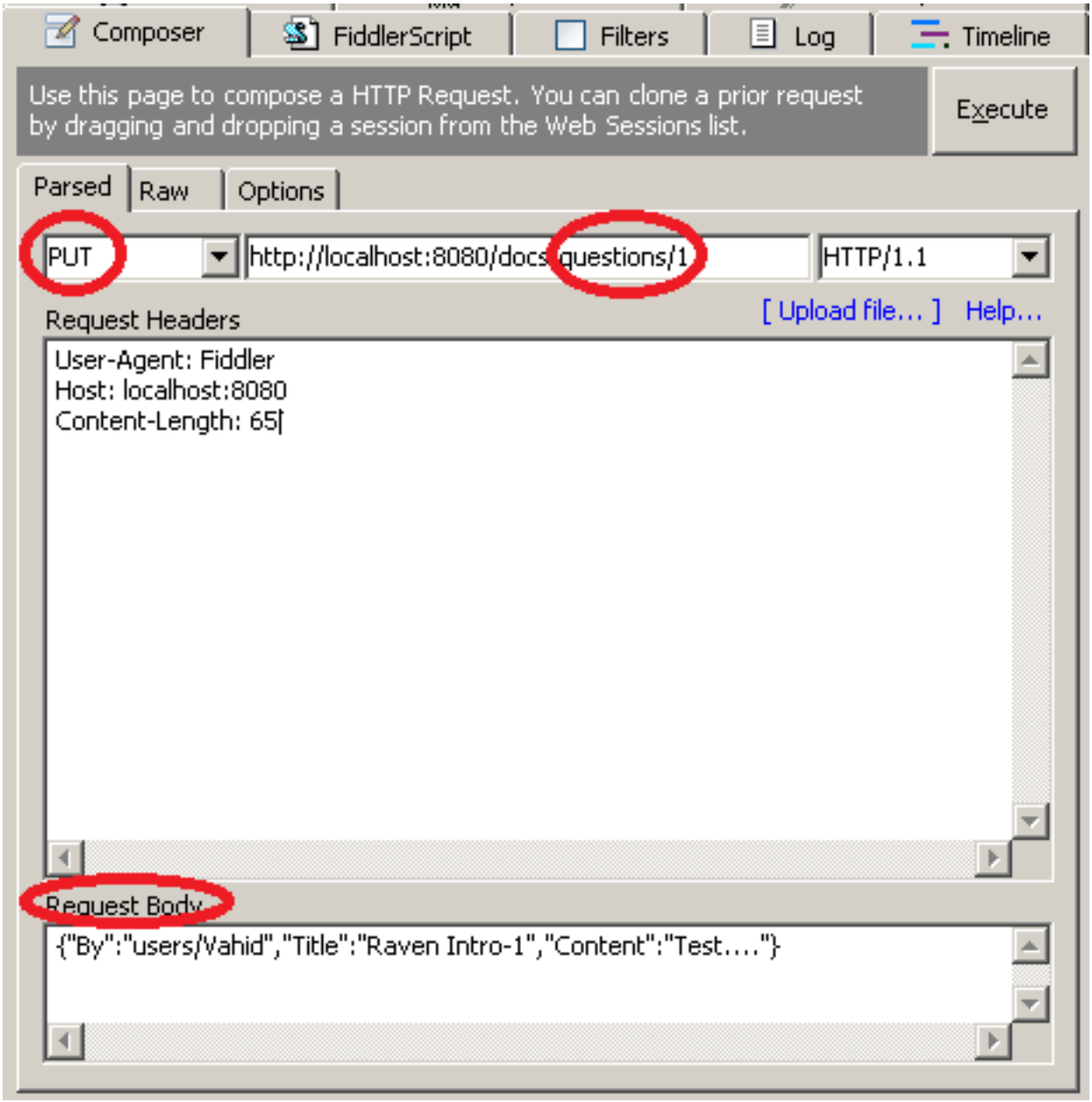


به علاوه در اینجا یک سری هدر اضافی (یا متادیتا) را هم می‌توان مشاهده کرد که RavenDB جهت سهولت کار کلاینت خود ارسال کرده است:



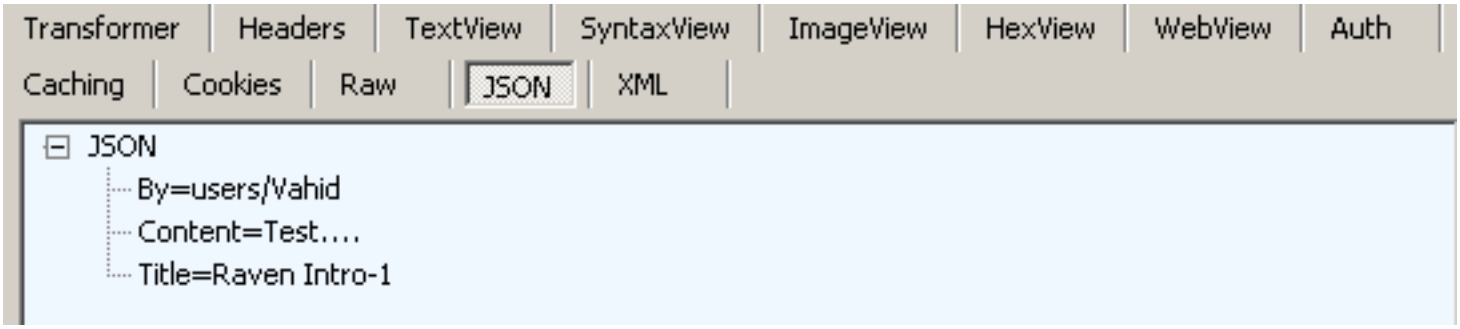
یک نکته: اگر آدرس `http://localhost:8080/docs/questions` را اجرا کنید، به معنای درخواست دریافت تمام سؤالات است. اما RavenDB به صورت پیش فرض طوری طراحی شده‌است که تمام اطلاعات را بازگشت ندهد و شعار آن `Safe by default` است. به این ترتیب مشکلات مصرف حافظه بیش از حد، پیش از بکارگیری یک سیستم در محیط کاری واقعی، توسط برنامه نویس یافت شده و مجبور خواهد شد تا برای نمایش تعداد زیادی رکورد، حتما صفحه بندی اطلاعات را پیاده سازی کرده و هربار تعداد معقولی از رکوردها را واکشی نماید.

کار با دستور PUT



اینبار نوع دستور را به PUT و آدرس را به http://localhost:8080/docs/questions/1 تنظیم می‌کنیم. همچنین در قسمت Request body، مقداری را که قرار است در سؤال یک درج شود، با فرمت JSON وارد می‌کنیم.

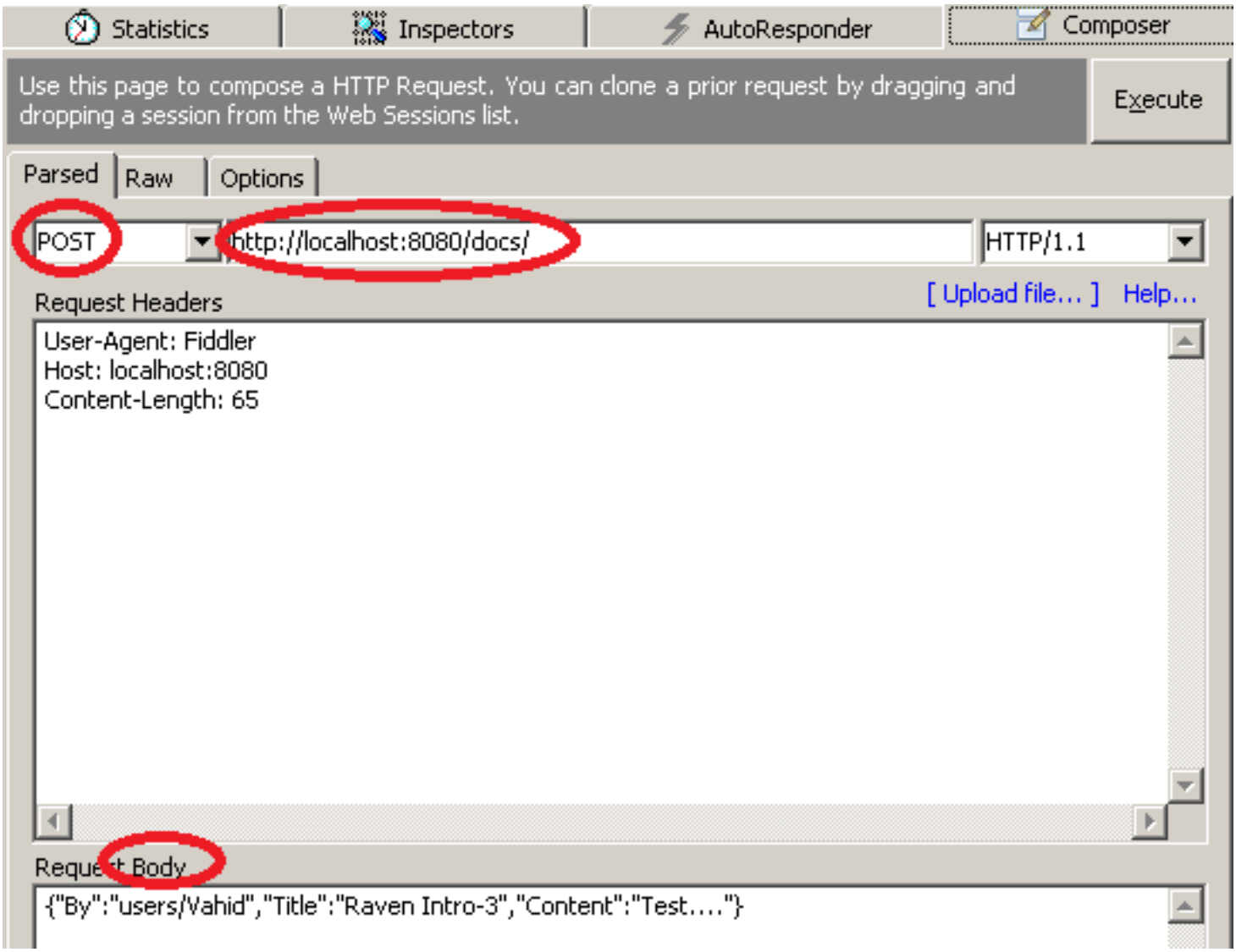
برای آزمایش صحت عملکرد آن، مرحله کار با دستور GET را یکبار دیگر تکرار نمائید:



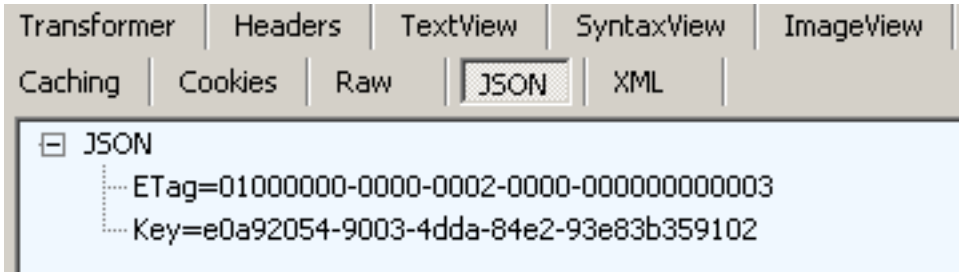
همانطور که مشاهده می‌کنید، تغییر ما در عنوان سؤال یک، با موفقیت اعمال شده است.

کار با دستور POST

در حین کار با دستور PUT، نیاز است حتما Id سؤال مورد نظر برای به روز رسانی (و یا حتی ایجاد نمونه جدید، در صورت عدم وجود) ذکر شود. اگر نیاز است اطلاعاتی به سیستم اضافه شوند و Id آن توسط RavenDB انتساب داده شود، بجای دستور PUT از دستور POST استفاده خواهیم کرد.



مطابق تصویر، اطلاعات شیء مدنظر را با فرمت JSON به آدرس http://localhost:8080/docs/ ارسال خواهیم کرد. در این حالت اگر به برگه‌ی Inspectors مراجعه نمائیم، یک چنین خروجی JSON ایی دریافت می‌گردد:



Key در اینجا شماره منحصربفرد سند ایجاد شده است و برای دریافت آن تنها کافی است که دستور GET را بر روی آدرس زیر که نمایانگر Key دریافتی است، اجرا کنیم:

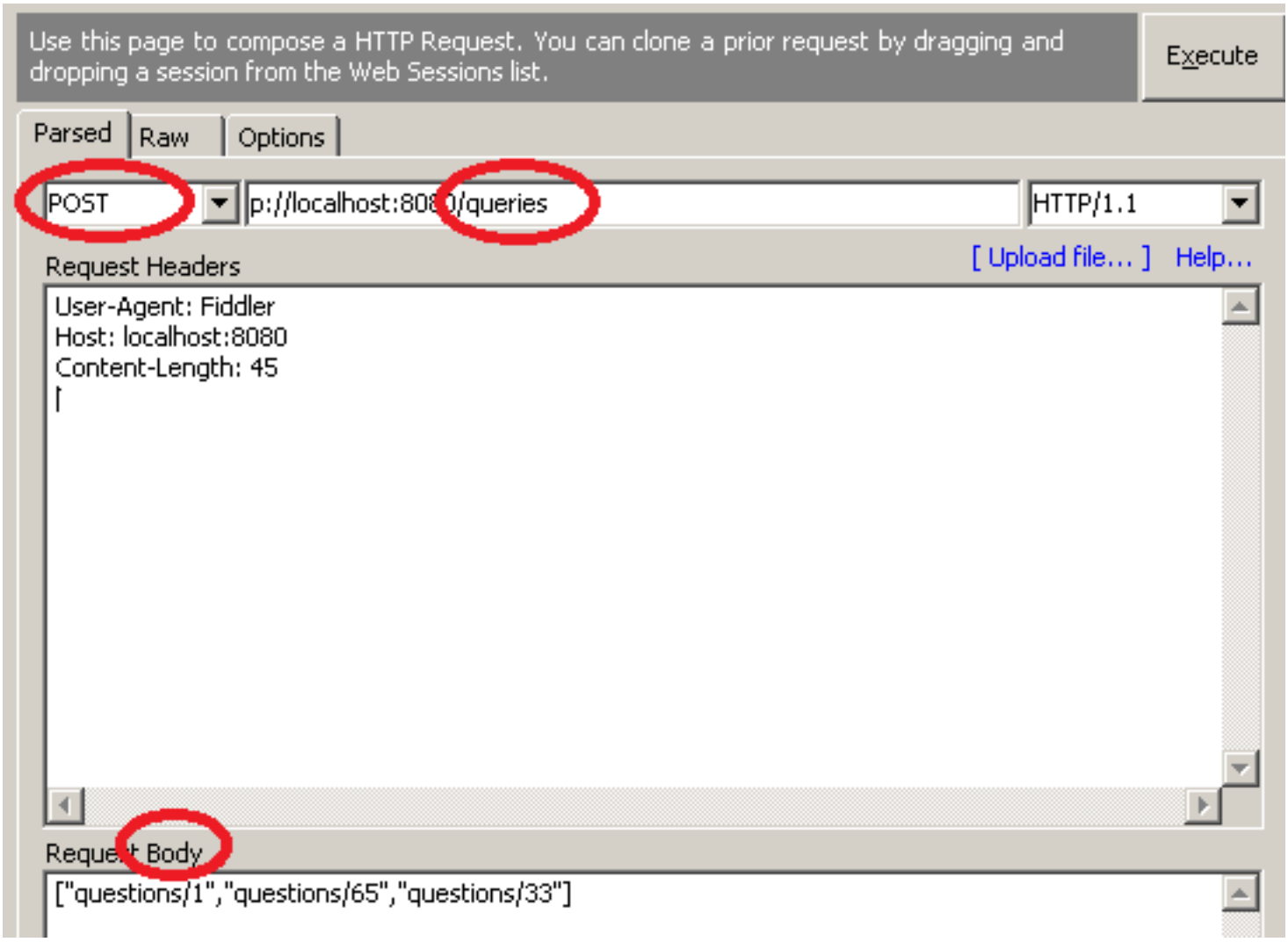
http://localhost:8080/docs/e0a92054-9003-4dda-84e2-93e83b359102

کار با دستور DELETE

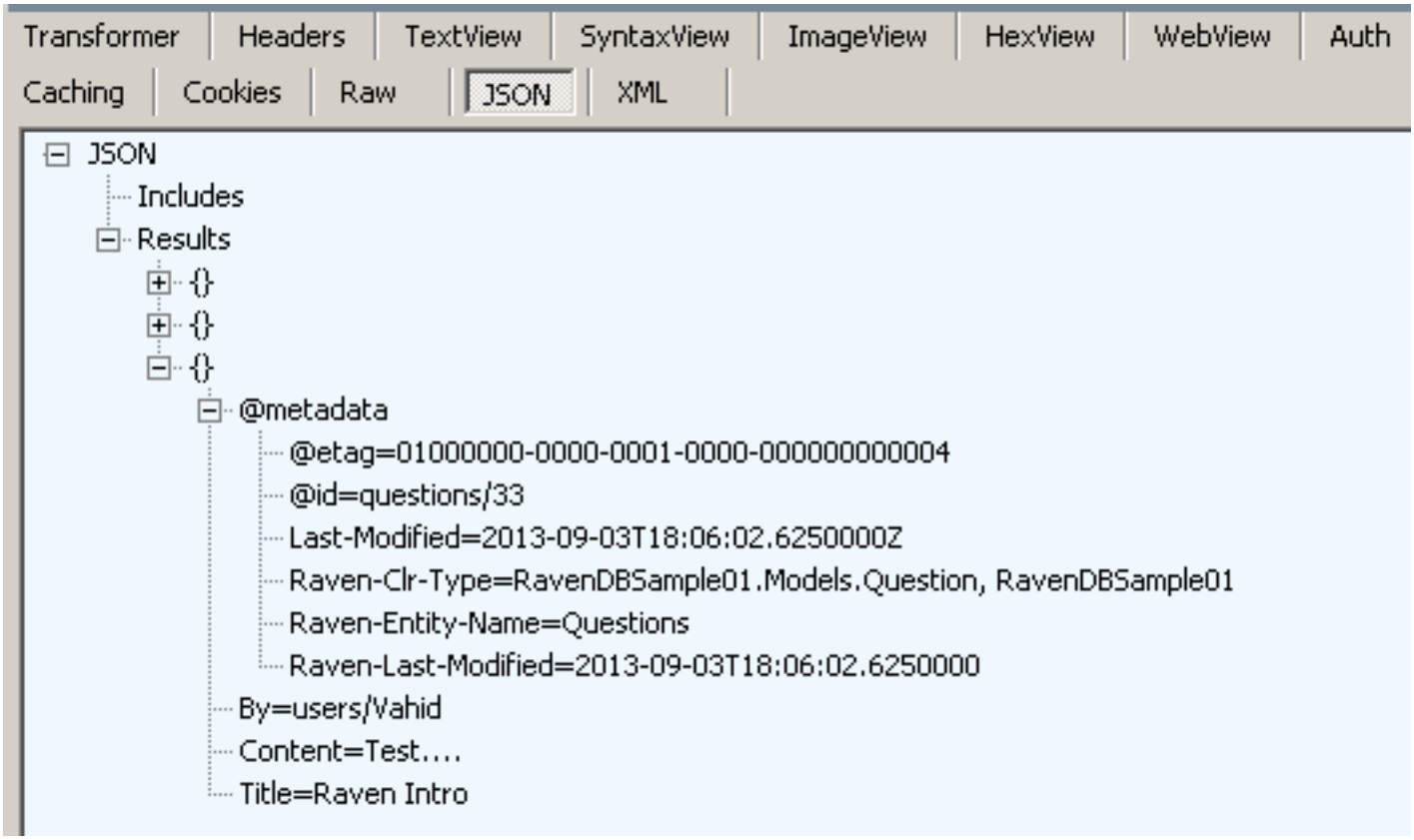
برای حذف یک سند تنها کافی است آدرس آن‌را وارد کرده و نوع دستور را بر روی Delete قرار دهیم. برای مثال اگر دستور Delete را بر روی آدرس فوق که به همراه Id تولید شده توسط RavenDB است اجرا کنیم، بلافاصله سند از بانک اطلاعاتی حذف خواهد شد.

بازگشت چندین سند از بانک اطلاعاتی RavenDB

برای نمونه، در فراخوانی‌های Ajaxایی نیاز است چندین رکورد با هم بازگشت داده شوند. برای این منظور باید یک درخواست Post ویژه را مهیا کرد:



در اینجا آدرس ارسال اطلاعات، آدرس خاص `http://localhost:8080/queries` است. اطلاعات ارسالی به آن، آرایه‌ای از Idهای اسنادی است که به اطلاعات آن‌ها نیاز داریم.



بنابراین برای کار با RavenDB در برنامه‌های وب و خصوصا کدهای سمت کلاینت آن، نیازی به کلاینت یا کتابخانه ویژه‌ای نیست و تنها کافی است یک درخواست Ajax از نوع post را به آدرس کوئری‌های سرور RavenDB ارسال کنیم تا نتیجه نهایی را به شکل JSON دریافت نمائیم.

نظرات خوانندگان

نویسنده: شهروز جعفری
تاریخ: ۱۳۹۲/۰۷/۲۳ ۱۴:۳۰

چرا در post از http://localhost:8080/docs/ استفاده شده و questions در url دیده نمیشه ولی در put و get دیده میشه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۷/۲۳ ۱۵:۲۱

put کار به روز رسانی و یا حتی ایجاد یک id مشخص رو انجام می‌دهد. اگر id مشخص نیست، از post استفاده خواهد شد تا محاسبه آن id خودکار شده و به سیستم واگذار شود.

نویسنده: رضا ساکت
تاریخ: ۱۳۹۲/۰۸/۱۶ ۲۱:۲۴

سلام

با این وصف کسی که نشانی سرور دیتابیس را داشته باشد مستقیم و به راحتی می‌تواند اطلاعات را دستکاری نماید. همینطور است؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۸/۱۶ ۲۱:۳۶

خیر. این دسترسی‌ها در این مثال میسر شد چون حالت پیش فرض نصب آزمایشی سورس باز آن، حالت دسترسی ادمین است .

نویسنده: رضا ساکت
تاریخ: ۱۳۹۲/۰۸/۱۶ ۲۱:۵۹

بنابراین برای کار ایمن در RavenDB میبایست آنرا خرید

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۸/۱۶ ۲۲:۶

- بله.
- البته می‌شود پورت‌های دسترسی خارجی به یک سرور را با فایروال بست. به این ترتیب فقط برنامه نصب شده در آن سرور امکان اتصال را خواهد داشت (خیلی‌ها با SQL Server هم به همین نحو کار می‌کنند؛ یک برنامه وب و یک برنامه سرور SQL دارند روی یک سرور. برنامه وب سفارشی، لایه اتصال امن به بانک اطلاعاتی است).
- همچنین حالت نصب embedded آن دسترسی از بیرون ندارد و فقط از طریق برنامه قابل استفاده است.