عنوان: **NoSQL ؟**
نویسنده: سروش ترک زاده
تاریخ: ۱۳۹۱/۰۴/۱۱ ۱۸:۲۳
آدرس: www.dotnettips.info
برچسب‌ها: JSON, NoSQL, RavenDB

به شما خواننده گرامی پیشنهاد می‌کنم مطلب قبلی " آشنایی با JSON؛ ساده - خوانا - کم حجم " که پیش درآمدی بر این موضوع است را مطالعه کنید.

NoSQL یک مفهوم عام است و تعریف ساده آن "پایگاه داده بدون SQL است". به این معنی که در آن خبری از جدول ها، روابط بین آن‌ها و ... نیست!
اما چرا باید با وجود اینکه SQL به اغلب نیازهای ما پاسخ داده است، باید سراغ تکنولوژی‌های دیگر رفت؟

وقتی نگاهی به لیست شرکت‌های بزرگی می‌اندازیم که جز مشتریان پر و پا قرص NoSQL هستند(+ و +)، تعجب می‌کنیم! آیا آن‌ها از قدرت و قابلیت‌های SQL بی خبراند؟

پاسخ این گونه از سوال‌ها به تحلیل سیستم مربوط می‌شود. به عهده تحلیل گر است تا با توجه به اجزاء سیستم و ارتباط آن‌ها بهترین روش را برای ذخیره سازی اطلاعات انتخاب کند.
NoSQL بر اساس نحوه پیاده سازی اش دسته بندی شده است؛ که مهم‌ترین آن‌ها در زیر آمده است :
Wide Column Store

Document Store

Key Value / Tuple Store

Graph Databases

Multimodel Databases

Object Databases

برای آشنایی بهتر با هر کدام به nosql-database.org مراجعه کنید.

انتخاب روش؛ یک مثال ساده :
فرض کنید روال استخدام نیروی کار جدید در یک سازمان، از قرار زیر باشد:

ثبت مشخصات فردی

ارائه مدارک تحصیلی

شرکت در آزمون استخدامی
شرکت در مصاحبه (درصورت قبول شدن در آزمون)
شرکت در دوره آموزشی (در صورت قبول شدن در مصاحبه)

روش‌های ممکن برای نگهداری اطلاعات :
روش اول، تهیه پوشه هایی برای نگهداری اطلاعات مربوط به هر مرحله به صورت مجزا است.

روش دوم، تهیه یک پرونده برای هر شخص و نگهداری اسناد مربوط به شخص (در هر مرحله) است.

انتخاب روش اول امکان پذیر است، اما باعث پیچیده‌تر شدن سیستم و اتلاف زمان می‌شود که مطلوب نیست. برای پیاده سازی روش دوم، SQL پاسخ گوی نیاز پروژه نیست و با توجه به نیاز پروژه بهترین روش نگهداری اطلاعات، Document Store (نگهداری اطلاعات بر اساس ساختار اسناد) است.
خوش بختانه تعداد پایگاه‌های داده ای که بر اساس تکنولوژی Document Store پیاده سازی شده اند، زیاد است و از قدرتمندترین آن‌ها می‌توان به CouchDB ، MongoDB و RavenDB اشاره کرد. هرکدام از این انتخاب‌ها مزایا و معایبی دارند که باید با توجه به نیاز خود، مقایسه ای انجام داده و بهترین را انتخاب کنید.
انتخاب من RavenDB بوده است و دلایل آن :
بر اساس زبان سی شارپ نوشته شده است و همچنین با LINQ خیلی خوب کار می‌کند.

Transaction را پشتیبانی می‌کند.

اساس ذخیره سازی آن JSON است.

محیط Management Studio کاربر پسندی دارد.

نقطه آغازین بحث بعد RavenDB خواهد بود که *Bryan Wheeler* (مدیر توسعه بسترهای نرم افزاری در msn) در باره آن گفته :

RavenDB just rocked my world. It’s extremely approachable, even for non-database guys – it took me less than 30 minutes to get up and running

خوشحال می‌شوم، نظرات و تجربیات شما را در رابطه با NoSQL بدانم.

نظرات خوانندگان

نویسنده: RaminMjj
تاریخ: ۱۳۹۱/۰۴/۱۱ ۱۸:۵۴

سلام.
ابتدا تشکر میکنم از مطلبی که دارید ارائه میدهید.
شیوه نگارشتون هم بسیار خوبه.
فقط یک پیشنهاد دارم. اونهم اینه که یک مطلب اختصاص بدهید به؛ در چه پروژه‌هایی باید از NoSQL استفاده کرد و چه پروژه‌هایی نباید.

نویسنده: mze666
تاریخ: ۱۳۹۱/۰۴/۱۱ ۱۹:۰

سلام - خیلی ممنون بابت مطلب خوبتون. فقط اگر براتون ممکنه یه آموزش گام به گام یا یه نمونه پروژه از RavenDB که به نظرم بهترین هستش رو بذارید. ممنون

نویسنده: مجتبی چنانی
تاریخ: ۱۳۹۱/۰۴/۱۱ ۱۹:۲۰

با سلام

من به عنوان کسی که در پروژه‌های خود از انوع ذخیره سازی‌ها بر اساس نیاز استفاده کردم(سرعت! راحتی! پلتفرم ها! و...) هم نظر میدم و هم پاسخ شما دوست عزیز را می‌دم.

قطعا انتخاب اینکه از چه روشی برای ذخیره سازی داده‌ها استفاده شود بسته به تیم پیاده سازکننده پروژه و نیز طراحان و... دارد. من با یک مثال توضیحی را خدمت شما می‌دهم.

در یک پروژه که اخیرا در حال اجرا هست(در دست من و هم تیمی‌های من) این پروژه یک پروژه بزرگ و با دیدها و اهداف وسیعی هست. ما در این برنامه هم از ادرس دهی بر اساس پوشه‌ها و دایرکتوری‌ها داده‌ها را ذخیره کردیم(اطلاعاتی مانند لینک فایل‌ها و یا تصاویر و...) و حتی در بعضی محل‌ها نیاز بود که اطلاعات یک فرد را در یک فایل xml قرار می‌دادیم و بعضی وقتها هم در پایگاه داده و هم فایل xml به این دلیل که در مورد اول تنها برنامه سمت کلاینت نیاز به این اطلاعات داشت و در آنجا پارسر قوی xml وجود داشت اما در مورد دوم ما به یک سری دیتا نیاز داشتیم که هم در سرور به آنها نیاز داریم و هم کلاینت! خب در بحث وب ما به مدیران اگر می‌خواستیم xml ارائه کنیم قطعا راه حل خوبی نبود و از سرعت و کارایی ما کم می‌کرد لذا از پایگاه داده استفاده کردم ولی برای زمانی که کاربر کلاینتی ما نیاز به اطلاعات داشت به این دلیل که بار سرور زیاد نشود از xmlها استفاده می‌شد که با یک لینک مستقیم می‌توانست به دست اورد(البته خود لینک همین فایل xml هم ساخته می‌شد! هیچ جا ذخیره نمی‌شد!)

عذر می‌خوام اگر بجای نویسنده پاسخ دادم البته این پاسخ من خیلی سربسته بود و انشا.. مفید بوده.

از نویسنده مطلب بابت مطلب خوبشون که کم دیدم در تارنماهای فارسی به اون بپردازن(متاسفانه بسیاری از اساتید دانشگاهی با این مفهوم حتی اشنایی ندارند با اینکه دانستن کلیت ان یک تعریف ساده است!) موفق باشید.

نویسنده: سروش ترک زاده
تاریخ: ۱۳۹۱/۰۴/۱۱ ۱۹:۲۷

از شما ممنونم به خاطر پیشنهاد خوبتون.

به نظر خودم موضوعی که شما مطرح کردید جای بحث بیشتری دارد و حتما این مورد رو برای نوشته‌های آینده مد نظر قرار خواهم داد.

نویسنده: سروش ترک زاده
تاریخ: ۱۳۹۱/۰۴/۱۱ ۱۹:۳۰

ممنون از شما که این مثال رو مطرح کردید.
در نوشته‌های من جای یک مثال برای واضح شدن بیشتر موضوع خالی بود!

نویسنده: سروش ترک زاده
تاریخ: ۱۳۹۱/۰۴/۱۱ ۱۹:۳۶

موافقم، هیچ چیز مثل یک مثال کاربردی یادگرفتنی نیست...

نویسنده: امیرحسین جلوداری
تاریخ: ۱۳۹۱/۰۴/۱۱ ۱۹:۴۰

یه مشکلی داره RavenDb ! ... اونم اینه که مجوزش از هموناس که اگه پروژه تجاری باشه باس پولشو بدی! (اگه متن باز که باشه هیچ!)

نویسنده: mze666
تاریخ: ۱۳۹۱/۰۴/۱۱ ۱۹:۴۷

بله درسته ولی اگه بخوایم حساب کنیم ما از خیلی چیزا توی برنامه‌های تجاریمون استفاده میکنیم (Telerik, Stimulsoft, ...) ولی پولشو نمیدیم. اینم روش. (البته نمیگم کار خوبی مکینیم!)

نویسنده: RaminMjj
تاریخ: ۱۳۹۱/۰۴/۱۱ ۲۲:۱۱

با تشکر از جوابی که دادید.
ولی من میخواستم مطلبی مشابه این مقاله ارائه بشه تا بیشتر با NoSQL آشنا بشیم
http://www.dbta.com/Articles/Editorial/Trends-and-Applications/SQL-or-NoSQL-How-to-Choose-the-Right-Database-for-Your-Application-71240.aspx

نویسنده: peyman
تاریخ: ۱۳۹۱/۰۴/۱۱ ۲۲:۲۶

فکر میکنم Neo4j هم عالی باشه ! گر چه مایکروسافت هم داره روی Trinity کار میکنه که هر دو از نوع گراف دیتابیس‌ها هستن و به نظر من کار با گراف دیتابیس‌ها خیلی زیباتر و لذتبخش‌تر هست

نویسنده: محمد
تاریخ: ۱۳۹۱/۰۴/۱۱ ۲۲:۵۷

اتلاف زمان صحیح است و نه «اطلاف زمان». اتلاف از تلف کردن میاد. ممنون به هر حال.

نویسنده: ghafoori
تاریخ: ۱۳۹۱/۰۴/۱۱ ۲۳:۱۵

اگر برنامه داخل ایران باشه اینکارو می‌کنیم اما اگر بخواهیم اون را داخل سرور امریکا یا هر دیتاسنتری که به مجوزها گیر میده برنامه را داشته باشیم باید چکار کنیم اینجا مانگو خودش را بهتر نشون میده

نویسنده: نیما
تاریخ: ۱۳۹۱/۰۴/۱۲ ۱:۴

سلام دوست عزیز

ممنون از مطلبتون. در نگاه اول مطلب شما اینجوری به من القا کرد که اطلاعات مثلا سریالایز بشن حالا چه بصورت json و یا xml . من رو پروژه ای کار کردم که اطلاعات بصورت xml بود و فرض کنید حجم این فایلهای xml به حدود 10 کیلو بطور میانگین میرسید . گزارشگیری از این xmlها بسیار وقت گیر بود مخصوصا که اگر قرار بود group by یا اعمال دیگری رو انجام بدیم و خیلی اوقات به timeout میخورد که با عوض کردن این شیوه و قرار دادن اطلاعات در جداول مختلف مشکلات بکلی حل شد. ممنون میشم بیشتر توضیح بدین. موفق باشید

نویسنده: رضا.ب
تاریخ: ۱۳۹۱/۰۴/۱۲ ۲:۱۴

دوست عزیز، لطفا بیشتر توضیح دهید. من چند بار کامنت‌تون رو خوندم متوجه نشدم چی میگین. ممنون.

نویسنده: رضا.ب
تاریخ: ۱۳۹۱/۰۴/۱۲ ۳:۱۱

دو سوال داشتم:
- امکان انتقال (Migrate) بین یه دیتابیس relational و nosql در عمل ممکن هست؟ (منظورم تبدیل رابطه‌ای‌ها به nosqlهاست. چون برعکسش محاله ظاهرا؟!)
- نقش ORMها در برقراری ارتباط Objectی و منطق برنامه‌های شی‌گراء با این نوع دیتابیس‌های براساس سند(بدون ساختار) کجاست؟ اصلا ORM معنی میده هنگام کار با NoSQL؟
با تشکر. ممنونم.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۱۲ ۸:۴۰

- در ravendb امکان replication به sql server وجود دارد.
- یکی از اهداف مهم ORMها در دات نت، نوشتن کوئری‌های strongly typed است. در ravendb شما از روز اول با کوئری‌های strongly typed سروکار دارید. همچنین از همان ابتدای کار هم با کلاس‌های دات نتی و نگاشت خودکار آن‌ها کار می‌کنید. کلا ravendb برمبنای معماری و همچنین توانمندی و پیشرفت‌های زبان‌های دات نتی تهیه شده.

نویسنده: mze666
تاریخ: ۱۳۹۱/۰۴/۱۲ ۹:۴۴

سلام آقای نصیری - میخواستم از شما یا آقای ترک زاده خواهش کنم که یه مورد از اینها (RavenDB, CouchDB, ...) که به نظر خودتون خوبه رو آموزش بدید.
یه آموزش اساسی مثل MVC یا Code First که تو چند جلسه تمام مباحثش رو گفتید.
ممنون.

نویسنده: سروش ترک زاده
تاریخ: ۱۳۹۱/۰۴/۱۲ ۱۰:۳۹

ما توی مکتب این جوری گفته بودن بهمون...
ممنون که تزکر دادین. اسلاح شد :-)

نویسنده: سروش ترک زاده
تاریخ: ۱۳۹۱/۰۴/۱۲ ۱۰:۵۶

ممنون از جناب آقای نصیری که پاسخشان در رابطه با ORM کامل و کافی بود.
اما در مورد سوال اول شما :
در بعضی موارد تبدیل پایگاه داده Table-Relational به بعضی موارد مثل Document Store کاملا امکان پذیر است؛ اما تبدیل آن به نوع KeyValue اساسا معنی ندارد، زیرا کاربرد این دو روش کاملا متفاوت است.
اما این نکته قابل توجه است که اگر تحلیل سیستم شما بر اساس Table-Relational انجام گرفته باشد؛ بعد از تبدیل به Document-Store، با کاهش سرعت مواجه می‌شوید.
و به نظر من زمانی باید سراغ روش‌های NoSQL رفت که ساختار Table-Relational پاسخ مناسبی برای نیاز ما نباشد.

نویسنده: سروش ترک زاده
تاریخ: ۱۳۹۱/۰۴/۱۲ ۱۱:۱۳

سلام
نظر شما تا حدودی صحیح است اما کلاس‌های دات نت مثل XMLWriter , XDocument و ... قابل مقایسه با Engine قدرتمندی که برای یک پایگاه داده نوشته می‌شود، نیستند.
همچنین یکی از نیازها که باعث می‌شود سراغ NoSQL برویم، حجم عظیم اطلاعات است.
پس هیچ نگرانی در مورد حجم اطلاعات نباید وجود داشته باشد...

نویسنده: رضا.ب
تاریخ: ۱۳۹۱/۰۴/۱۲ ۱۵:۳۷

خب یعنی برای رفتن سمت هر NoSQLی باید دلیل مرجحی داشته باشیم. و بخاطر جدید بودن و استفاده سازمان‌های عظیم از آنها و یا حتی آسان‌تر بودن، دلیل نمیشود که پایگاه‌داده‌ای رابطه‌ای رو رها کنیم.
و این زمانی اتفاق میوفته که این 6 نوعی که ذکر کردید، رو کاملا بشناسیم. مزایا معایب و موارد کاربرد اون‌رو بدونیم و با اثبات ردِ کارایی مطلوب دیتابیس‌های رابطه‌ای به انتخاب NoSQLی دست بگذاریم.
در مورد کامنتتون متوجه نشدم علت اینکه یه پایگاه داده‌ی رابطه‌ای چرا نمیتونه به جفت مقدار/کلید تبدیل بشه؟ شاید بلعکس‌اش محال باشه. مثلا وراثت یا جدول‌های با ستون‌های پویا و ... اصلا در پایگاه‌های رابطه‌ای بی‌معنی هستند.

نویسنده: سروش ترک زاده
تاریخ: ۱۳۹۱/۰۴/۱۳ ۱۵:۵

شاید من نتونستم منظور خودم رو واضح بگم؛
Table-Relational و NoSQL نقطه مقابل هم نیستند و انتخاب شما بین یکی از روش‌های ذخیره کردن اطلاعات (Graph Databases ، Object Databases ، Table Relational و ...) مشابه مثال انتخاب یکی از Type هایی مثل bit ، TimeSpam ، long و ... برای ذخیره کردن یک مقدار کوچک است. درست است که همه این کارها را با string می‌توان انجام داد و لی می‌توان با انتخاب درست در سرعت و فضایی که قرار است مصرف شود، صرفه جویی کرد.

و در باره مورد بعد که مطرح کردید، شاید یک مثال ساده قضیه رو روشن‌تر کند؛ می‌شود یک عدد کوچک رو در متغییری از جنس TimeSpam ریخت، اما اگر این عدد به معنی زمان نباشد، روش ما بهینه و حتی درست نیست، اما کار انجام شده است...
در صورتی که می‌شود این مقدار را در یک متغییر از جنس int ذخیره کرد.

امیدوارم شبهه ای که برای شما ایجاد شده است، با ارائه یک مثال کاربردی از RavenDB که در پست بعدی خواهم گفت، برطرف شود...

نویسنده: محمد صاحب
تاریخ: ۱۳۹۱/۰۴/۱۴ ۱۲:۴۰

تا آماده شدن مثال کاربردی; دیدن این پست خالی از لطف نیست.

نویسنده: محسن
تاریخ: ۱۳۹۱/۱۰/۲۳ ۱۲:۱۰

سلام

از RavenDB راضی بودین؟ آیا واقعا از جستجوی Full-Text بهره مند است و تونسته Lucene رو خوب تعبیه کنه؟

نویسنده: سروش ترک زاده
تاریخ: ۱۳۹۱/۱۰/۲۵ ۱۳:۳۰

سلام
تا اندازه ای که کارکردم خوب بود، البته پیش نیومد که توی پروژه Enterprise از آن استفاده کنم و در مورد Full-Text , Lucene راستش تا حالا امتحان نکردم...
شاید دوستان دیگر بتوانند راهنمایی کنند.

نویسنده: بازرگان
تاریخ: ۱۳۹۱/۱۱/۲۰ ۱۴:۴۴

گوگل پلاس و فیسبوک برای بانک اطلاعاتشون از چه شیوه هایی استفاده میکنند ؟

نویسنده: سعید
تاریخ: ۱۳۹۱/۱۱/۲۰ ۱۷:۲۱

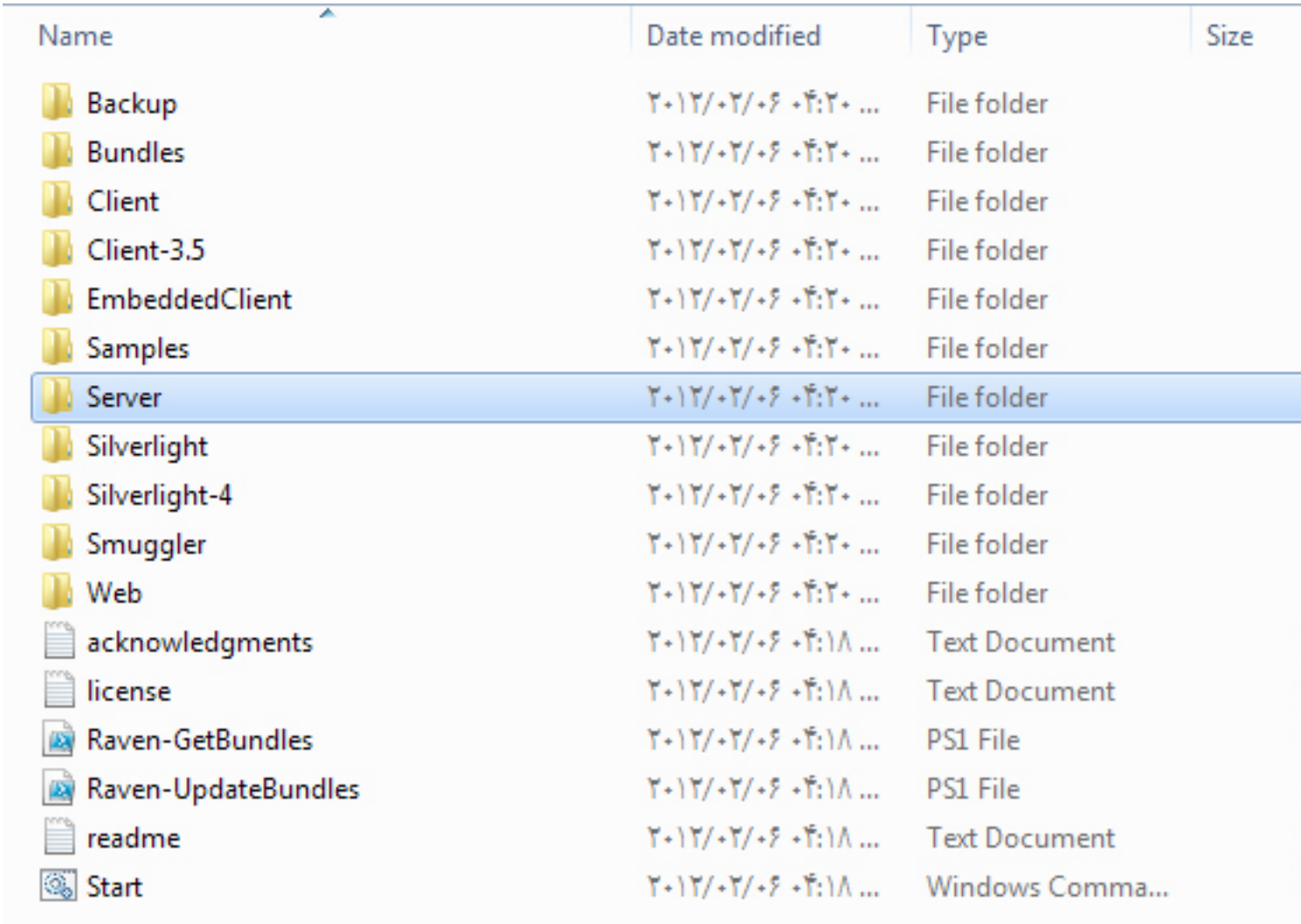
گوگل از بانک اطلاعاتی ساخت خودش استفاده می‌کنه: اطلاعات بیشتر ، فیس بوک هم در اینجا

عنوان: **RavenDB؛ تجربه متفاوت از پایگاه داده**
نویسنده: سروش ترک زاده
تاریخ: ۱۳۹۱/۰۴/۱۵ ۱۹:۳۱
آدرس: www.dotnettips.info
برچسب‌ها: C#, JSON, NoSQL, RavenDB

" به شما خواننده گرامی پیشنهاد می‌کنم مطلب قبلی را مطالعه کنید تا پیش زمینه مناسبی در باره این مطلب کسب کنید. "

ماهیت این پایگاه داده وب سرویسی مبتنی بر REST است و فرمت اطلاعاتی که از سرور دریافت می‌شود، JSON است.

گام اول: باید آخرین نسخه RavenDB را دریافت کنید. همان طور که مشاهده می‌کنید، ویرایش‌های مختلف کتابخانه هایی که برای نسخه Client و همچنین Server طراحی شده است، دراین فایل قرار گرفته است.



برای راه اندازی Server باید فایل Start را اجرا کنید، چند ثانیه بعد محیط مدیریتی آن را در مرورگر خود مشاهده می‌کنید. در بالای صفحه روی لینک Databases کلیک کنید و در صفحه باز شده گزینه New Database را انتخاب کنید. با دادن یک نام دلخواه حالا شما یک پایگاه داده ایجاد کرده اید. تا همین جا دست نگه دارید و اجازه دهید با این محیط دوست داشتنی و قابلیت‌های آن بعدا آشنا شویم.

در گام دوم به Visual Studio می‌رویم و نحوه ارتباط با پایگاه داده و استفاده از دستورات آن را فرا می‌گیریم.

گام دوم:

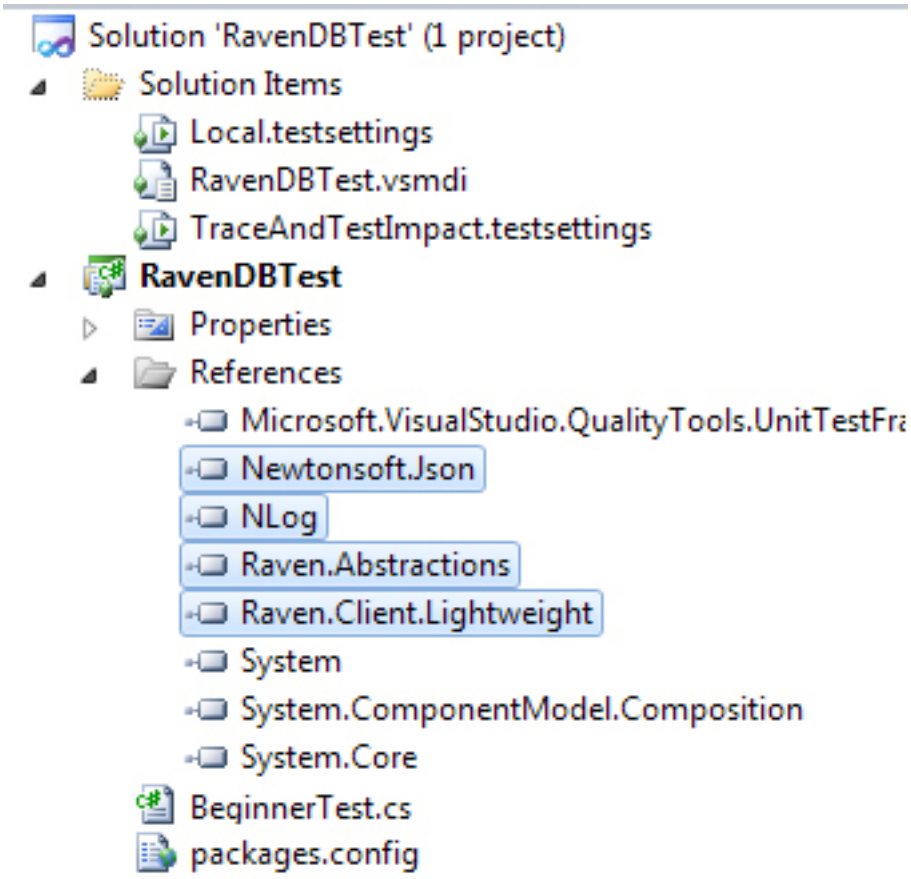
با یک پروژه Test شروع می‌کنیم که در هر گام تکمیل می‌شود و می‌توانید پروژه کامل را در پایان این پست دانلود کنید.

برای استفاده از کتابخانه‌های مورد نیاز دو راه وجود دارد:
استفاده از NuGet : با استفاده از دستور زیر Package مورد نیاز به پروژه شما افزوده می‌شود.

```
PM> Install-Package RavenDB -Version 1.0.919
```

اضافه کردن کتابخانه‌ها به صورت دستی : کتابخانه‌های مورد نیاز شما در همان فایلی که دانلود شده بود و در پوشه Client قرار دارند.

کتابخانه هایی را که NuGet به پروژه من اضافه کرد، در تصویر زیر مشاهده می‌کنید :



با Newtonsoft.Json در اولین بخش بحث آشنا شدید. NLog هم یک کتابخانه قوی و مستقل برای مدیریت Log است که این پایگاه داده از آن بهره برده است.

" دلیل اینکه از پروژه تست استفاده کردم ؛ تمرکز روی کدها و مشاهده تاثیر آن‌ها ، مستقل از UI و لایه‌های دیگر نرم افزار است. بدیهی است که استفاده از آن‌ها در هر پروژه امکان پذیر است. "

برای شروع نیاز به آدرس Server و نام پایگاه داده داریم که می‌توانید در App.config به عنوان تنظیمات نرم افزار شما ذخیره شود و هنگام اجرای نرم افزار مقدار آن‌ها را خوانده و در متغییرهای readonly ذخیره شوند.

```
<appSettings>
    <add key="ServerName" value="http://SorousH-HP:8080/"/>
    <add key="DatabaseName" value="TestDatabase" />
</appSettings>
```

هنگامی که صفحه Management Studio در مرورگر باز است، می‌توانید از نوار آدرس مرورگر خود آدرس سرور را به دست آورید.

```
[TestClass]
    public class BeginnerTest
    {
        private readonly string serverName;
        private readonly string databaseName;

        public BeginnerTest()
        {
            serverName = ConfigurationManager.AppSettings["ServerName"];
            databaseName = ConfigurationManager.AppSettings["DatabaseName"];
        }
    }
```

برای برقراری ارتباط با پایگاه داده نیاز به یک شئ از جنس DocumentStore و جهت انجام عملیات مختلف (ذخیره، حذف و ...) نیاز به یک شئ از جنس IDocumentSession است. کد زیر، نحوه کار با آن‌ها را به شما نشان می‌دهد :

```
[TestClass]
    public class BeginnerTest
    {
        private readonly string serverName;
        private readonly string databaseName;

        private DocumentStore documentStore;
        private IDocumentSession session;

        public BeginnerTest()
        {
            serverName = ConfigurationManager.AppSettings["ServerName"];
            databaseName = ConfigurationManager.AppSettings["DatabaseName"];
        }

        [TestInitialize]
        public void TestStart()
        {
            documentStore = new DocumentStore { Url = serverName };
            documentStore.Initialize();
            session = documentStore.OpenSession(databaseName);

        }

        [TestCleanup]
        public void TestEnd()
        {
             session.SaveChanges();
             documentStore.Dispose();
             session.Dispose();
        }
    }
```

در طراحی این پایگاه داده از اگوی Unit Of Work استفاده شده است. به این معنی که تمام تغییرات در حافظه ذخیره می‌شوند و به محض اجرای دستور ;()session.SaveChanges ارتباط برقرار شده و تمام تغییرات ذخیره خواهند شد.

هنگام شروع (تابع : TestStart) متغییر session مقدار دهی می‌شود و در پایان کار (تابع : TestEnd) تغییرات ذخیره شده و منابعی که توسط این دو شئ در حافظه استفاده شده است، رها می‌شود.

البته بر مبنای طراحی شما، دستور ;()session.SaveChanges می‌تواند پس از انجام هر عملیات اجرا شود.

برای آشنا شدن با نحوه ذخیره کردن اطلاعات، به کد زیر دقت کنید:

```
class User
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string Address { get; set; }
        public int Zip { get; set; }اهی}
    }
```

```
[TestMethod]
        public void Insert()
        {
            var user = new User
                            {
                                Id = 1,
                                Name = "John Doe",
                                Address = "no-address",
                                Zip = 65826
                            };
            session.Store(user);
        }
```

اگر همه چیز درست پیش رفته باشد، وقتی به محیط RavenDB Studio که هنوز در مرورگر شما باز است، نگاهی می‌اندازید، یک سند جدید ایجاد شده است که با کلیک روی آن، اطلاعات آن قابل مشاهده است.
لحظه‌ی لذت بخشی است...
یکی از روش‌های خواندن اطلاعات هم به صورت زیر است:

```
[TestMethod]
        public void Select()
        {
            var user = session.Load<User>(1);
        }
```

نتیجه خروجی این دستور هم یک شئ از جنس کلاس User است.

تا این جا، ساده‌ترین مثال‌های ممکن را مشاهده کردید و حتما در بحث بعد مثال‌های جالب‌تر و دقیق‌تری را بررسی می‌کنیم و همچنین نگاهی به جزئیات طراحی و قراردادهای از پیش تعیین شده می‌اندازیم.

" به شما پیشنهاد می‌کنم که منتظر بحث بعدی نباشید! همین حالا دست به کار شوید... "

نسخه بدون کتابخانه‌های موردنیاز (2 مگابایت) : RavenDBTest_Small.zip
نسخه کامل (15 مگابایت) : RavenDBTest.zip

نظرات خوانندگان

نویسنده: ناصر طاهری
تاریخ: ۱۳۹۱/۰۴/۱۶ ۱۷:۱۲

سلام.
مقوله‌ی جالبیه برای من.
منتظر ادامه هستم. موفق باشید.

نویسنده: حسین
تاریخ: ۱۳۹۱/۰۴/۱۶ ۱۹:۱۳

بسیار ممنون که کاربردی پیش میرین.من پروژمو با همین روش پیاده سازی می‌کنم و از اطلاعات خوبتون استفاده کردم.

نویسنده: ramin_rp
تاریخ: ۱۳۹۱/۰۴/۱۷ ۱۰:۴۳

سلام
وقتی ravendb رو استارت میکنم و محیط مدیریت اون تو browser اجرا میشه برای انجام عملیاتی مثل ایجاد دیتابیس user,pass میخواد که هرچی میدم قبول نمیکنه
حتی raven رو به عنوان service هم نصب کردم ولی مشکل حل نشد
جستجو تو نت هم نتیجه نداد
مشکل از چیه؟

(مستندات رسمی ravendb خوب نیست، مستندات mongodb واقعا کامل و جامع هست)

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۱۷ ۱۱:۳۷

توضیحات بیشتر در اینجا

```
By default RavenDB allow anonymous access only for read requests (HTTP GET),
and since we creating data, we need to specify a username and password.
You can control this by changing the AnonymousAccess setting in the server configuration file.
Enter your username and password of your Windows account and a sample data will be generated for you.
```

نویسنده: سروش ترک زاده
تاریخ: ۱۳۹۱/۰۴/۱۷ ۲۱:۱۸

سلام
ببخشید که دیر به سوال شما پاسخ دادم...
یه راه دیگه، علاوه بر راهی که توسط جناب آقای نصیری ارائه شده است، وجود دارد.
در پوشه Server فایل Raven.Server.exe را با Notepad باز کنید، سپس مقدار تنظیمات با کلید "Raven/AnonymousAccess " را به "All" تغییر دهید.توجه کنید که به بزرگ و کوچک بودن حروف حساس است.

در ضمن RavenDB از نظر سابقه و تعداد کاربران، قابل مقایسه با پایگاه داده هایی مثل SQL نیست و حق با شماست...

نویسنده: صابر

تاریخ: ۱۳۹۱/۰۴/۲۱ ۱۱:۱۹

سلام
نمی‌دانم مشکل از چیه ؟ ولی وقتی من سعی می‌کنم که بسته‌ی RavenDB رو از طریق Nuget دریافت کنم Error زیر رو میده .

```
Install-Package : The element 'metadata' in namespace
'http://schemas.microsoft.com/packaging/2010/07/nuspec.xsd' has invalid child element
'frameworkAssemblies' in namespace
'http://schemas.microsoft.com/packaging/2010/07/nuspec.xsd'. List of possible elements expected:
'summary' in namespace
'http://schemas.microsoft.com/packaging/2010/07/nuspec.xsd'.
At line:1 char:1
+ Install-Package RavenDB -Version 1.0.919
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
+ CategoryInfo : NotSpecified: (:) [Install-Package], InvalidOperationException
+ FullyQualifiedErrorId : NuGet.VisualStudio.Cmdlets.InstallPackageCmdlet
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۲۱ ۱۱:۵۸

به احتمال زیاد VS.NET شما دسترسی به اینترنت ندارد (^ و ^).

نویسنده: peyman
تاریخ: ۱۳۹۱/۰۵/۰۸ ۹:۱۸

آقا سروش کی شروع میکنی سری جدید رو منتظریم قربان !

نویسنده: سروش ترک زاده
تاریخ: ۱۳۹۱/۰۵/۰۸ ۱۸:۵۹

سلام
ببخشید که دیر شد، به احتمال زیاد پنجشنبه ادامه آموزش را روی سایت قرار خواهم داد...

نویسنده: پژمان
تاریخ: ۱۳۹۱/۰۶/۰۱ ۲۳:۲۵

ravendb هم مثل اینکه برای جستجو از لوسین استفاده می‌کنه.

نویسنده: فرزاد
تاریخ: ۱۳۹۱/۰۷/۱۰ ۲۳:۵۰

سلام
میخوام یه نرم افزار تحت ویندوز بنویسم که نیاز دارم از بانک اطلاعاتی استفاده کنم از طرفی هم نمی‌خوام با Sql server و یا access کار کنم چون نیاز به نزم افزار هایی با حجم زیاد هست که برای کاربر دردسر میشه.
میخواستم اگه میشه بفرمایید که به نظرتون از چی استفاده کنم بهتره؟
ممنون

نویسنده: حسین مرادی نیا
تاریخ: ۱۳۹۱/۰۷/۱۱ ۵:۴۸

نکته اینکه وقتی بانک اطلاعات Access رو استفاده کنین ، حتما نیازی نیست که Access روی کامپیوتر کاربر نصب باشه تا بتونه از برنامه شما استفاده کنه.
به هر حال میتونید از Sql Server CE استفاده کنید: http://www.dotnettips.info/search/label/SQL%20Server%20CE

عنوان: **استفاده از درایوها در Window Azure Storage جهت استفاده در RavenDB**
نویسنده: سام ناصری
تاریخ: ۱۳۹۲/۰۱/۲۶ ۲:۵
آدرس: www.dotnettips.info
برچسب‌ها: RavenDB

در تلاش برای راه اندازی دیتابیس RavenDB بر روی Windows Azure چند مقاله ای خوندم که گاهی خیلی گیج کننده بود. الان تقریباً به نتایجی رسیده‌ام و دوست دارم در این مقاله نکاتی رو که به نظرم دانستن آنها بایسته است را مطرح کنم. باشد که مفید واقع شود.

پیش زمینه 1، یکی دیگر از روشهای راه اندازی RavenDB:

راه اندازی سرویس، نصب بر روی IIS و استفاده به صورت توکار، روش‌هایی هستند که در خود مستندات نچندان کامل RavenDB در حال حاضر مطرح شده است. راه دیگری که برای راه اندازی RavenDB می‌تواند مورد استفاده قرار گیرد، از طریق برنامه نویسی است. یعنی سرور RavenDB را با اجرای کد بالا می‌آوریم. نگران نباشید، این کار خیلی سخت نیست و به سادگی از طریق نمونه سازی از کلاس HttpServer و ارائه پارامترهای پیکره‌بندی و فراخوانی یک و یا دو متود می‌تواند صورت گیرد. مزیت این روش در پویایی و انعطاف پذیری آن است. شما می‌توانید هر تعداد سرور را با هر پیکره‌بندی پویایی، بالا بیاورید.

به کلمه HttpServer خوب دقت کنید. بله، درست است؛ این یک سرور کامل است و تمام درخواست‌های Http را طبق قواعد RavenDB و البته HTTP پاسخ می‌دهد. حتی studio ی RavenDB, که یک برنامه Silverlight است, نیز سرو میشود. (برنامه Silverlight در ریسورسهای RavenDB.Database.dll توکار(embed) شده است.)

کد مینیمالیست نمونه، یک RavenDB http server در قالب یک برنامه Console Application:

```
static void Main(string[] args)
{
    var configuration = new Raven.Database.Config.RavenConfiguration() {
        AccessControlAllowMethods = "All",
        AnonymousUserAccessMode = Raven.Database.Server.AnonymousUserAccessMode.All,
        DataDirectory = @"C:\Sam\labs\HttpServerData",
        Port = 8071,
    };
    var database = new Raven.Database.DocumentDatabase(configuration);
    var server = new Raven.Database.Server.HttpServer(configuration, database);
    database.SpinBackgroundWorkers();
    server.StartListening();

    Console.WriteLine("RavenDB http server is running ...");
    Console.ReadLine();
}
```

با اجرای برنامه فوق، پایگاه داده شما در پورت 8071 ماشین، فعال است و آماده پاسخگویی. استودیوی RavenDB نیز از طریق مسیر http://127.0.0.1:8071 قابل دسترسی است.

چرا این مطلب را گفتم، چون برای راه اندازی RavenDB در Azure می‌خواهیم از این روش استفاده کنیم. در یک worker role دیگر ما نه IIS داریم و نه یک virtual machine در اختیار داریم تا یک service را بر روی آن نصب کنیم. پس بهترین گزینه برای ما راه اندازی سرور RavenDB از طریق برنامه نویسی است.

پیش زمینه 2، چندساکنی در RavenDB و مسیر داده ها:(Multi Tenancy)

یک سرور RavenDB می‌تواند چندین پایگاه داده را میزبانی کند. هر چند به طور پیش فرض تک ساکنی برگزیده شده است. اما شما می‌توانید پایگاه‌های داده جدید را به سیستم اضافه کنید. مشکلی که من با مستندات RavenDB دارم این است که به طور پیش فرض درباره زمانی مصداق پیدا می‌کنند که RavenDB در حالت تک ساکنی مورد استفاده قرار میگیرد.

مهم است که بدانید مسیری که به عنوان مسیر داده‌ها در هنگام راه اندازی سرور ارائه می‌دهید برای پایگاه داده پیش فرض مورد استفاده قرار میگیرد و باید مسیرهای جداگانه مستقلی برای پایگاه داده‌های بعدی تنظیم کنید.

توجه داشته باشید که در RavenDB اگر در هنگام ساخت پایگاه داده، مسیری را مطرح نکنید، مسیر پیش فرض انتخاب خواهد شد. همچنین در حالت چندساکنی هم هیچ ارتباطی بین پایگاه‌های داده بعدی با پایگاه داده <system> وجود ندارد و همواره مسیر پیش

فرض به صورت Databases/dbName/~ خواهد بود که dbName نام پایگاه داده مورد نظر شما است. مهم است که بدانید که ~ در مسیر فوق دارای تعریف رسمی ای نیست و آنچه از کد بر می‌آید ~ مسیر BaseDirectory برای AppDomain جاری است. پس با توجه اینکه نوع برنامه میزبان سرور چیست (IIS, Windows Service, Worker Role) مقدار آن می‌تواند متفاوت باشد.

تعریف Worker Role برای RavenDB

در واقع مطلب اصلی درباره نحوه استفاده از CloudDrive در Web Role یا Worker Role است. همانطور که میدانید Web Role و Worker Role هر دو برای ذخیره سازی داده‌ها مناسب نیستند. در واقع بایستی با این رویکرد به آنها نگاه کنید که فقط کدهای اجرایی بر روی آنها قرار بگیرند و نه چیز دیگری. در مورد استفاده پایگاه داده RavenDB در Windows Azure می‌توانید آن را به صورت یک Worker Role تعریف کنید. اما برای اینکه داده‌ها را ذخیره کنید بایستی از یک Cloud Drive استفاده کنید.

خوب، در ابتد لازم است که کمی درباره‌ی CloudDrive بدانیم؛ خواندن این مطلب درباره‌ی اولین انتشار Windows Azure Drive خالی از لطف نیست.
حالا برای اینکه RavenDB را راه بیاندازیم باید نخست Wroker Role را بسازیم و سپس قطعه کدی بنویسیم تا درایو مجزا و مختصی را برای اینکه RavenDB اطلاعات را در آن بریزد بسازد. در آخر باید Worker Role را تنظیم کنیم تا درایو ساخته شده را در خود mount کند.

برای ساختن درایو قطعه کد زیر آن را انجام میدهد:

```
CloudStorageAccount storageAccount = CloudStorageAccount.FromConfigurationSetting(connectionString);
// here is when later on you may add code for inititalizing CloudDrive chache
CloudBlobClient blobClient = storageAccount.CreateCloudBlobClient();
blobClient.GetContainerReference("drives").CreateIfNotExist();

CloudDrive cloudDrive = storageAccount.CreateCloudDrive(
    blobClient
    .GetContainerReference("drives")
    .GetPageBlobReference("ravendb4.vhd")
    .Uri.ToString()
);

try
{
    // create a 1GB Virtual Hard Drive
    cloudDrive.Create(1024);
}
catch (CloudDriveException /*ex*/ )
{
    // the most likely exception here is ERROR_BLOB_ALREADY_EXISTS
    // exception is also thrown if the drive already exists
}
```

در کد فوق نامهای drives و ravendb.vhd کاملاً اختیاری هستند. اما باید از قواعد نامگذاری container پیروی کنند.
برای سوار کردن درایو قطعه کد زیر آن را انجام میدهد:

```
string driveLetter = cloudDrive.Mount(25, DriveMountOptions.Force);
```

توجه داشته باشید که کد سوار کردن درایو، قاعدتاً، بایستی در Worker Role صورت بگیرد و همچنین باید قبل از راه اندازی RavenDB باشد.
این یک ایراد طراحی Windows Azure است که شما نمیتوانید حرف درایو را خودتان انتخاب کنید، بلکه خروجی متود Mount مشخص میکند که درایو در چه حرف درایوی سوار شده است. و شما محدود هستند که کدهای خود را به گونه ای بنویسید که مسیر ذخیره سازی اطلاعات در Cloud Drive را ثابت فرض نکند و ارجاعات به این مسیرها شامل حرف درایو نباشد.

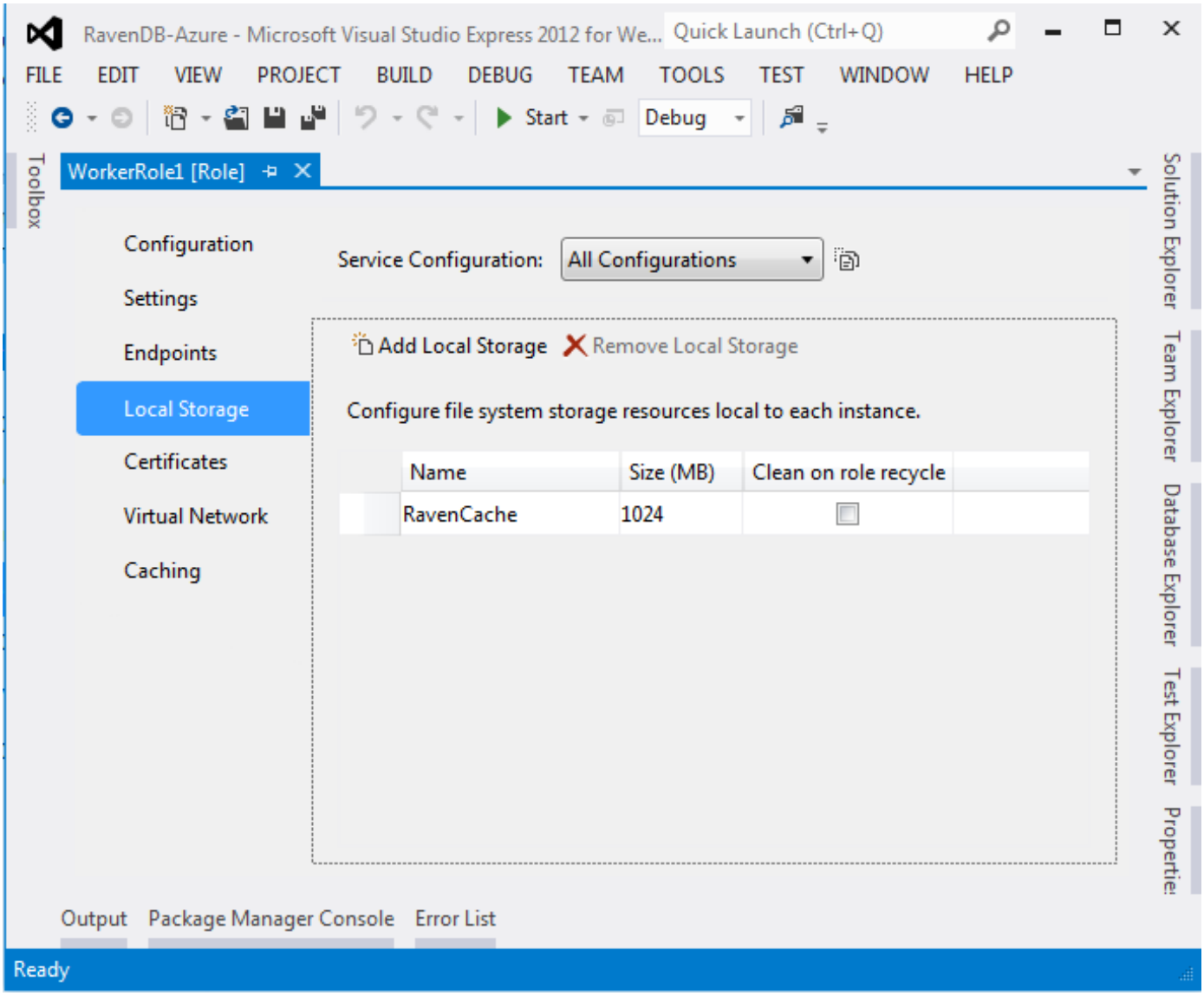
رفع مشکل کندی درایو در Windows Azure با تعریف کش:

کد فوق برای راه اندازی درایو مورد نظر ما کافی است. اما هنوز دارای یک مشکل اساسی و مهم است و آن اینست که بسیار کند عمل خواهد کرد.

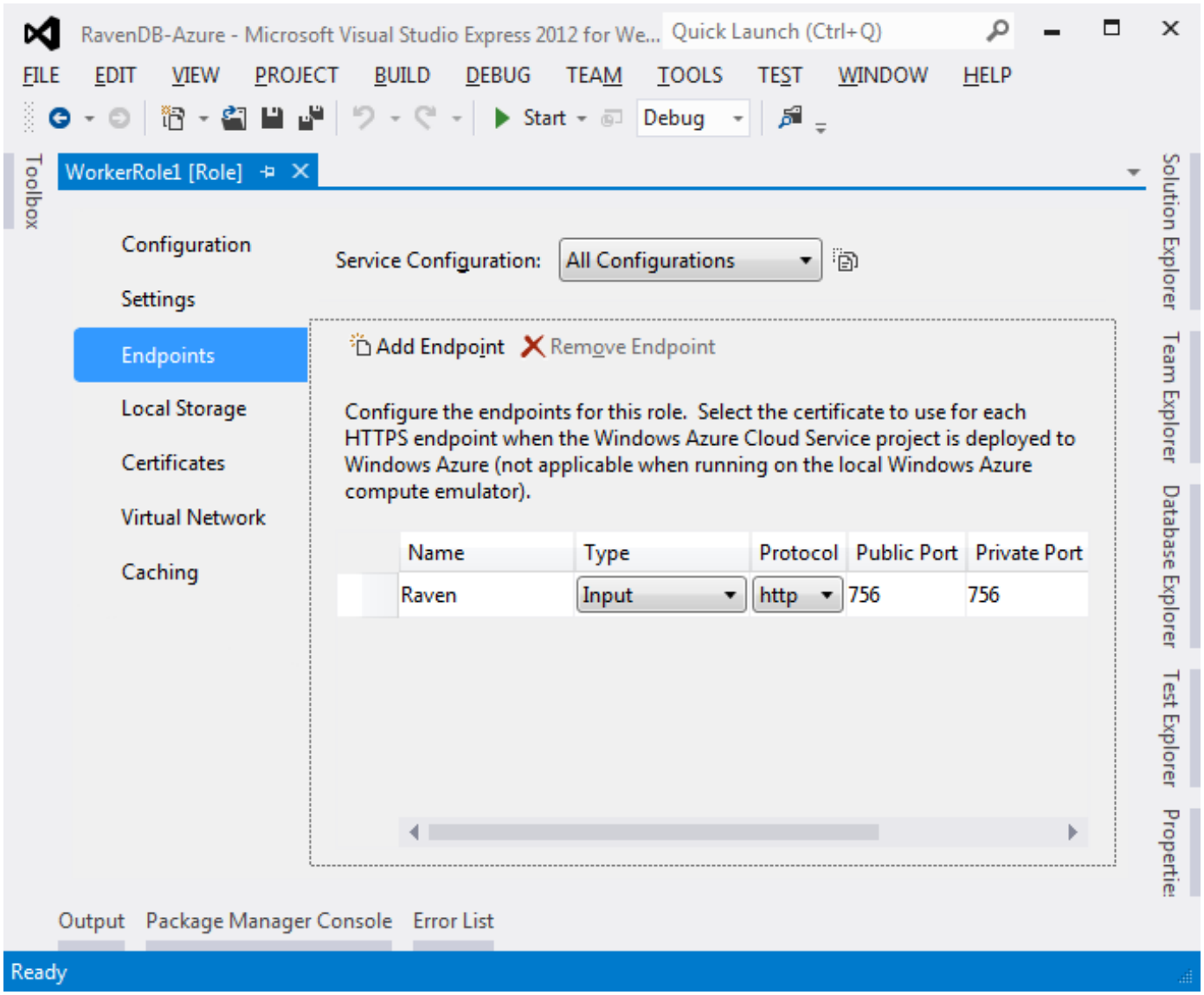
با فراخوانی متود CloudDrive.InitializeCache این متود به طور اتوماتیک برای تمام درایوهای mount شده یک کش محلی فراهم میکند و در نتیجه network I/O کمتری صورت خواهد گرفت. توجه داشته باشید که در صورت استفاده از این متود بایستی کش را برای Worker Role تعریف کنید. در صورت عدم استفاده از این متود کارائی پایگاه داده شما به شدت افت میکند. کد زیر را قبل از تعریف هر نوع درایوی قرار دهید.

```
LocalResource localCache = RoleEnvironment.GetLocalResource("RavenCache");
CloudDrive.InitializeCache(localCache.RootPath, localCache.MaximumSizeInMegabytes);
```

در کد فوق RavenCache نام یک Local Storage است که شما در تنظیمات Worker Role تعریف میکنید.(نام آن اختیاری است.) برای تعریف Local Storage بایستی در قسمت تنظیمات Worker Role رفته و آنگاه زبانه Local Storage رفته و سپس یک Local Storage را به مانند تصویر زیر اضافه کنید. نام که میتواند هر نامی باشد. اندازه را به اندازه مجموع درایوهایی که میخواهید در Worker Role تعریف کنید قرار دهید(در مثال برنامه ما در اینجا مقدار 1024) و گزینه Clean on role recycle را آنتیک کنید.



حال که درایو مورد نیاز ما آماده است قدم دیگر این است که پورتی را که RavenDB میخواهد در آن فعال شود را تعریف کنیم.
برای اینکار بایستی در قسمت تنظیمات Worker Role در زبانه Endpoints رفته و یک endpoint جدید به آن مطابق تصویر زیر ارائه کنیم.



حال که پورت هم تنظیم شده است میتوانیم RavenDB را در Worker Role راه بیاندازیم:

```
var config = new RavenConfiguration
{
    DataDirectory = driveLetter,
    AnonymousUserAccessMode = AnonymousUserAccessMode.All,
    HttpCompression = true,
    DefaultStorageTypeName = "munin",
    Port = RoleEnvironment.CurrentRoleInstance.InstanceEndpoints["Raven"].IPEndpoint.Port,
    PluginsDirectory = "plugins"
};

try
{
    documentDatabase = new DocumentDatabase(config);
    documentDatabase.SpinBackgroundWorkers();
    httpServer = new HttpServer(config, documentDatabase);
```

```
    try
    {
        httpServer.StartListening();
    }
    catch (Exception ex)
    {
        Trace.WriteLine("StartRaven Error: " + ex.ToString(), "Error");

        if (httpServer != null)
        {
            httpServer.Dispose();
            httpServer = null;
        }
    }
}
catch (Exception ex)
{
    Trace.WriteLine("StartRaven Error: " + ex.ToString(), "Error");

    if (documentDatabase != null)
    {
        documentDatabase.Dispose();
        documentDatabase = null;
    }
}
```