Couchbase with Windows and .NET - Part 1

In this three part series, we're going to look at the basics of interacting with Couchbase for .NET developers on Windows. We'll start with the basics, and build towards a "vertical" slice of a complete ASP.NET MVC app on the .NET 4.x framework. For a deeper dive, please check out my blog posts on Couchbase[1] and the Couchbase Developer Portal[2].

In this first part, we'll install Couchbase Server and go over the basics of how it works.

In the second part, we'll look at using ASP.NET with Couchbase Server.

In the final part, we'll implement all the CRUD functionality in an ASP.NET application.

1. Installing Couchbase Server

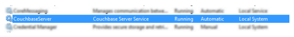
Let's start with the easiest part:

1.1. Download and install Couchbase Server[3].

Windows 10, Mac OS X and Linux packages are all available. Go to the Downloads page on the Couchbase website[4]. We'll use the Enterprise Edition for this blog post. Let's proceed with a Windows 10 installation.

Run the downloaded installation file, and go through a 4-step wizard.

At this point, Couchbase should be running as a Windows Service (verify by opening services.msc[5]).



CoreMessaging	Manages communication betwe...	Running	Automatic	Local Service
CouchbaseServer	Couchbase Server Service	Running	Automatic	Local System
Credential Manager	Provides secure storage and retri...	Running	Manual	Local System

We'll automatically be taken to the Couchbase Console via web browser to customize the Couchbase setup. We can always return to this console via http://localhost:8091.

1 http://blog.couchbase.com/
2 http://developer.couchbase.com
3 http://www.couchbase.com/nosql-databases/downloads
4 http://www.couchbase.com/nosql-databases/downloads
5 https://technet.microsoft.com/en-us/library/cc755249.aspx

Click "Setup" to start the configuration wizard. Let’s stick with the default settings (mostly) for now.

1. In step 1, we need to specify how much RAM to give to Couchbase. It might be a good idea to dial down some of the default RAM Quotas (we can always change them later). Since we only need the one node for development, let’s enabled all services on this node (Index, Query, and Full Text). I would recommend checking out some of the free Couchbase training[6] available to dive deeper into tweaking these settings.

1. In step 2, we can choose to install some sample data. For this blog post, we don’t need to do this. However, the "travel-sample" is useful for trying out some of the N1QL functionality (more on that later).
2. In step 3, let’s create a "default" bucket, to follow along with this blog post.
3. In step 4, we can elect to receive notifications from Couchbase and register the product.
4. Finally, in step 5, we need to enter a username and password to create an administrator account.

Now we’re ready to start using Couchbase. On the "overview" page, we can see how much RAM is available to Couchbase, and how much is actually in use. If we wanted to scale out Couchbase by adding servers, then we would see them listed in the Servers section. If we click on the "Data Buckets" tab, we’ll see that there’s at least the default bucket we created.

Feel free to play around with the Couchbase Console and check out the Couchbase Console documentation[7].

2. Some NoSQL lingo

But before we dive into some code, let’s go over some of the lingo with Couchbase. It’s not a difficult tool to use, but it is different from RDBMS systems like SQL Server.

Node

A node is an invidual machine that is running Couchbase Server.

Cluster

[6] http://learn.couchbase.com/
[7] http://developer.couchbase.com/documentation/server/4.5/admin/ui-intro.html

Nodes are joined together into a cluster[8] that act in concert to provide scaling and improved availability. Scaling: more nodes means more RAM and disk space. Availability: if one node goes down, the cluster will continue to function. As developers, we don’t generally have to worry about node management and clustering in code: it’s all handled by Couchbase and the Couchbase .NET SDK.

Bucket

A bucket[9] is a place to store documents. Each document has a key. Within a bucket, each key must be unique. Typically, a single bucket corresponds to a single application. The documents within a bucket do not have to be similar at all. We could store a document that contains information about a user, and a document with information about a building, both in the same bucket.

Document

In a basic sense, a Couchbase bucket is a giant `Dictionary<string,string>`. Each entry in a bucket consists of a key and a document. Documents consist of JSON. Because Couchbase uses JSON documents, it’s called a "document database".

N1QL

With Couchbase Server, we can write queries in a language called N1QL[10] (N1QL stands for "Non-first Normal Form Query Language, pronounced "nickel"). N1QL is a superset of SQL. This means that, basically, if you know SQL, then you know N1QL. An example:

```
SELECT name AS bookName, author AS bookAuthor
FROM `books-bucket`
WHERE YEAR(published) >= 1998
```

That will return something like:

```
{
 "results": [
```

[8] http://developer.couchbase.com/documentation/server/current/clustersetup/manage-cluster-intro.html
[9] http://developer.couchbase.com/documentation/server/4.5/clustersetup/bucket-setup.html
[10] http://developer.couchbase.com/documentation/server/4.5/developer-guide/querying.html

```
   { "bookName" : "The Little Book of Calm", "bookAuthor" : "Manny Bianco" },
   { "bookName" : "AOP in .NET", "bookAuthor" : "Matthew D. Groves" }
  ]
}
```

As we'll see later, the Linq2Couchbase library leverages N1QL into a LINQ provider that is similar to Entity Framework, NHibernate.Linq, etc.

Indexes

Indexes[11] in Couchbase are as important as in relational databases. Probably more so, because buckets contain a variety of documents.

To enable N1QL queries on a bucket, we need to at least create a primary index. This is an index on the bucket itself. Here's how to create one with N1QL: `` CREATE PRIMARY INDEX ON `my-bucket` ``

We can create indexes based on the fields in the documents. For instance, if we have a lot of documents that have "name" or "author" fields, and we end up querying based on this fields often, we can create indexes for them. These are "secondary indexes." As the size of a bucket grows, so does the importance of indexing.

3. See you later!

Stay tuned for the next two blog posts in this series. Please leave a comment, ping me on Twitter[12], or email me (matthew.groves AT couchbase DOT com). I'd love to hear from you.

[11] http://developer.couchbase.com/documentation/server/4.5/indexes/n1ql-in-couchbase.html
[12] http://twitter.com/mgroves