

# NoSQL and .NET

Using MongoDB and NoRM  
Hartford Code Camp  
2010-06-19

John C. Zablocki  
Development Lead, MagazineRadar  
Adjunct, Fairfield University

# Agenda

- What is NoSQL?
- NoSQL Databases
- Introducing MongoDB
- .NET Drivers for MongoDB
- Introducing NoRM
- Sample: Nerd Dinner on MongoDB
- Case Study: RateMySnippet.com
- Questions?

# NoSQL



# What is NoSQL?

- Coined in 1998 by Carlos Strozzi to describe a database that did not expose a SQL interface
- In 2008, Eric Evans reintroduced the term to describe the growing non-RDBMS movement
- Broadly refers to a set of data stores that do not use SQL or a relational model to store data
- Popularized by large web sites such as Google, Facebook and Digg

# Why NoSQL?

- Modern relational databases simply do not scale to Internet proportions
- Sites like Digg, Facebook and Ebay have data sets 10s or 100s of TB large.
- ACID support isn't required for blogs, status updates, product listings, etc.
- Reduce object-relational impedance

# NoSQL Databases

- NoSQL databases come in a variety of flavors
  - XML (myXMLDB, Tamino, Sedna)
  - Wide Column (Cassandra, Hbase, Big Table)
  - Key/Value (Redis, Memcached with BerkleyDB)
  - Graph (neo4j, InfoGrid)
  - Document store (CouchDB, MongoDB)

# No-Schema

- NoSQL databases generally do not impose a formal schema
- Optimized storage of unstructured data

# No-Schema Documents

- Related data is stored in a single document
- The ubiquitous Blog example
  - Each post is a document
  - Post might have reference to Blog or User document or store blog or user redundantly
  - Each post document has a nested collection of tags and comments
- Application layer must be smarter about referential integrity (eventual consistency)

# Introducing MongoDB

- Open source, document-oriented database
- 10gen corporate entity behind development
- 10gen supports official drivers for many platforms, but not .NET!

# MongoDB – The Basics

- Documents in a MongoDB database are stored in schema-less collections
- Documents are stored in BSON (Binary JSON)
- JavaScript used to query and manipulate collections
- Each item in a collection has a unique (primary) key called an ObjectId
- MongoDB uses a memory-mapped file where MRU documents are cached

# MongoDB – Advanced Features

- Text search
  - Incredibly fast with MongoDB
  - JavaScript regex searches are supported
- Indexes
  - Non-Primary and key indexes are supported
  - Unique key indexes are supported
- MapReduce (reuse recycle)
  - Used for aggregation and batch manipulation

# Installing MongoDB on Windows

- Download the binaries from [mongodb.org](http://mongodb.org)
- Extract to Program Files directory (or wherever)
- Create a directory `c:\data\db`
- Run `mongod.exe` from the command line with the `--install` switch
  - Check links at the end for a couple of gotchas
- To run the daemon without installing, simply run `mongod.exe` without arguments
- Run `mongo.exe` to verify the daemon is running

# MongoDB - Shell

- The MongoDB interactive JavaScript shell (`mongo.exe`) is a command line utility for working with MongoDB servers
- Allows for CRUD operations on collections
- May be used for basic administration
  - Creating indices
  - Cloning databases
- Also useful as a test-bed while building apps

# MongoDB – Shell Cntd.

- Connect to a server:port/database (defaults are localhost:27017/test ):
  - `mongo.exe localhost:27017/CodeCamp`
- Switch database:
  - `use AnotherDatabase`
- View collections in a database:
  - `show collections`

# MongoDB – Shell CRUD

- Insert an item into a collection
  - `db.Artists.insert({ Name : “The Shins” });`
- Find an item in a collection:
  - `db.Artists.findOne({ Name: “Radiohead”});`
- Find items in a collection:
  - `db.Artists.find({ Name : /The/i});`
- Count items in a collection
  - `db.Artists.count();`

# MongoDB – Shell CRUD Cntd.

- Update an item in a collection
  - `db.Artists.update({ Name : “JackJohnson” },  
$set : { Name : “Jack Johnson” } });`
- Update items in a collection
  - `db.Artists.update({ Name : { $ne : null } },  
{ $set : { Category : “Rock” } }, false, true);`
- \$ denotes special operators and operations
  - \$push, \$pop, \$pull, etc.

# .NET and MongoDB

- No officially supported 10gen driver
- mongo-csharp
  - Actively being developed and supported
  - Supports typed collections
- Simple-mongodb
  - Json-centric
  - Actively developed, but less so than other drivers like mongo-csharp and NoRM

# NoRM



# NoRM

- OK, it's Pronounced No R M!
- Actively developed by Andrew Theken, Adam Schroder, Karl Seguin and Rob Conery
- Active community with reliable support
  - I received help even as I prepared this slide!
- Support for typed and untyped collections, map/reduce and virtually all querying and updating operations and operators

# MongoDB – Shell CRUD

- Insert an item into a collection
  - `db.Artists.insert({ Name : “The Shins” })`
- Find an item in a collection:
  - `db.Artists.findOne({ Name: “Radiohead”});`
- Find items in a collection:
  - `db.Artists.find({ Name : /The/i});`
- Count items in a collection
  - `db.Artists.count();`

# NoRM - CRUD

- Connections managed with IDisposable pattern
  - using (Mongo mongo =  
Mongo.Create("connstring")) {  
//CRUD goes here  
}
- Mongo instance has MongoDBDatabase property
- MongoDBDatabase has GetCollection<T>  
methods for accessing MongoCollection
- CRUD operations performed on collections

# NoRM - CRUD

- Inserting a document into a typed collection
  - `mongo.Database.GetCollection<Artist>("Artists").insert(artist);`
- Updating a document in a typed collection
  - `mongo.Database.GetCollection<Artist>("Artists").UpdateOne(artist);`
- Updating a nested collection
  - `mongo.Database.UpdateOne(new { Name = "Radiohead"}, new { Albums = M.Push(album) });`

# NoRM – CRUD Cntd.

- Querying for all documents in a typed collection
  - `mongo.Database.GetCollection<Artist>().FindAll();`
- Querying with an Expression
  - `mong.Database.GetCollection<Artist>().FindOne(a => a.Name == "The Shins");`

# NoRM - MapReduce

```
MapReduce mr = mongo.Database.CreateMapReduce();  
    MapReduceOptions options = new  
MapReduceOptions("CollectionName"){ Map =  
mapFunction, Reduce = reduceFunction}  
    MapReduceResponse response = mr.Execute(options);  
    var collection =  
mongo.Database.GetCollection<MappedType>("OutputCo  
llectionName").AsQueryable();
```

# NoRM - LINQ

- LINQ provider exposed via `AsQueryable` method of `MongoCollection`
  - `mongo.Database.GetCollection<Artist>("Artists").AsQueryable();`
- Find items in typed collection
  - `var artistsStartingWithThe = from a in mongo.Database.GetCollection<Artist>().AsQueryable() where a.Name.Contains("The") select a`

# Example: Nerd Dinner

# Case Study: RateMySnippet.com

# Questions?

# More Info

- <http://dllHell.net> - my blog
- <http://www.CodeVoyeur.com> - my code
- <http://mongodb.org> - Official MongoDB site
- <http://groups.google.com/group/norm-mongodb>
- <http://www.summerofnorm.com> - Coming Soon!