# M101J: MongoDB for Java Developers

MongoDB University

Fall 2013

## Contents



## What is MongoDB?

- Lecture Video

### MongoDB is a non-relational JSON Document store.

This means that it does not support typical relational algebra or tables/columns/rows like your typical relational database. It stores documents in the JSON format.

Here is an example of JSON:

```
{
     "first_name": "Casey",
     "last_name": "Scarborough",
     "hobbies": ["guitar", "photography", "programming"]
}
```

MongoDB, unlike relational databases, is able to store documents that do not have the same structure.

Features that MongoDB doesn't have:

- Joins
- SQL
- Transactions

### Quiz

Which of the following statements are true about MongoDB? Check all that apply.

1. MongoDB is document oriented.
2. MongoDB supports joins.
3. MongoDB has dynamic schema.
4. MongoDB supports SQL.

- Answer Video

## MongoDB Relative to Relational DBs

- Lecture Video

MongoDB gives you a large depth of functionality while keeping high scalability and performance.

Documents are hierarchical and MongoDB doesn't support atomic transactions.

### Quiz

Which features did MongoDB omit in order to retain scalability?

1. Joins
2. Indexes
3. Secondary Indexes
4. Transactions across multiple selections

- Answer Video

# Building an App with MongoDB

- Lecture Video

## Application Architecture

The final application we are creating will be a blog using Java and MongoDB.

- MongoDB
- Starts a mongod process
- Java connects via TCP to MongoDB
- Java
- SparkJava
- FreeMarker
- MongoJava driver

*SparkJava* is a micro Java web framework inspired by Java that is inspired by Sinatra that easily allows you to creates routes and pages.

*FreeMarker* is a templating language.

# Introduction to Mongo Shell

- Lecture Video

After installing MongoDB, you can start the mongo shell by issuing the `mongo` command.

The first command you'll want to issue is to select the database:

```
> use test
```

This creates the database if it isn't created yet. Here are some common commands:

```
> db.things.save({ a : 1, b : 2, c : 3})
> db.things.find()
{ "_id" : ObjectId("52535e464a33ac3bdbb08566"), "a" : 1, "b" : 2, "c" : 3 }
> db.things.save({ a : 3, b : 2, c : 3, d : 200})
> db.things.find()
{ "_id" : ObjectId("52535e464a33ac3bdbb08566"), "a" : 1, "b" : 2, "c" : 3 }
{ "_id" : ObjectId("52535e724a33ac3bdbb08567"), "a" : 3, "b" : 2, "c" : 3, "d" : 200 }
> db.things.find({ a : 1 })
{ "_id" : ObjectId("52535e464a33ac3bdbb08566"), "a" : 1, "b" : 2, "c" : 3 }
```

## Quiz

Which of the following expressions are valid JSON documents? Check all that apply.

1. {a:1, b:2, c:3}
2. {a,1; b,4, c,6}
3. {a:1; b:1; c:4}
4. (A,1; b:2; c,4}

- Answer Video

## JSON Introduced

Issue the following commands in the Mongo Shell.

```
> db.test.save({a:1, b:1, fruits:['apple','orange','pear']})
> db.test.save({name:"casey", address:{street:"elm drive", city:"Morrow", zip:"30260", house_number: 655
> db.test.find().pretty()
{ "_id" : ObjectId("52535e2e4a33ac3bdbb08565"), "a" : 1 }
{ "_id" : ObjectId("52535e464a33ac3bdbb08566"), "a" : 1, "b" : 2, "c" : 3 }
{
  "_id" : ObjectId("52535e724a33ac3bdbb08567"),
  "a" : 1,
  "b" : 2,
  "c" : 3,
  "d" : 200
}
{
  "_id" : ObjectId("52535fb94a33ac3bdbb08568"),
  "a" : 1,
  "b" : 1,
  "fruits" : [
    "apple",
    "orange",
    "pear"
  ]
}
{
  "_id" : ObjectId("525360134a33ac3bdbb08569"),
  "name" : "casey",
  "address" : {
    "street" : "elm drive",
    "city" : "Morrow",
    "zip" : "30260",
    "house_number" : 6551
  }
}
```

### Quiz

Which of the following expressions are valid JSON documents? Check all that apply.

1. {a:1, b:2, c: 3}
2. {a:1, b:2, c:[1,2,3,4,5]}
3. {a:1, b:{}, c: [ { a:1, b:2}, 5, 6]}
4. { }

- Answer Video

# System Requirements

## Operating Systems

- Mac OS X 10.8
- Windows 7
- Linux

## Java

- Version 1.6
- Version 1.7

> *Note: You can check this by issuing the* `javac --version` *command from your terminal.*

# Installing MongoDB (Mac)

- Lecture Video

I installed MongoDB a little differently than the video does, and I'm listing that way. If you'd rather do it the way the video does, you can take a look there.

Issue the following commands from your home directory:

```
curl http://downloads.mongodb.org/osx/mongodb-osx-x86_64-2.4.6.tgz > mongodb.tgz
tar -zxvf mongodb.tgz
sudo mv mongodb-osx-x86_64-2.4.6 /usr/local/bin/mongodb
sudo mkdir -p /data/db
sudo chown `id -u` /data/db
export PATH=/usr/local/bin/mongodb/bin:$PATH
```

Or you can just install it through MacPorts.

```
port install mongodb
```

Then run the `mongod` command to start the server, and `mongo` to start the client.

## Installing and Using Maven

- Lecture Video

# The MongoDB Java Driver

- Lecture Video

In order to use Mongo from Java we need the Mongo Java Driver. We're going to add the following to our Maven pom.xml file:

```
<dependencies>
  <dependency>
    <groupId>org.mongodb</groupId>
    <artifactId>mongo-java-driver</artifactId>
    <version>2.10.1</version>
  </dependency>
</dependencies>
```

MongoDB has a database called course and a collection called hello, with the value `{ name: "MongoDB" }`. You can then create a new class and create a `public static void main()` function. Add the following to it:

```
import com.mongodb.MongoClient;
import java.net.UnknownHostException;

public class HelloWorldMongoDBStyle {

  public static void main(String[] args) {
    // Defaults to localhost, 27017
    MongoClient client = new MongoClient("localhost", 27017);

    // Get the database
    DB database = client.getDB("course");

    // Get the hello collection
    DBCollection = database.getCollection("hello");

    // Retrieve our document
    DBObject document = collection.findOne();
    System.out.println(document);
  }

}
```