MONGODB

1. **Install on OS X:**
2. **Install with homebrew**

Follow step:

* **brew update**
* **brew install Mongo DB**
* If have any error when download package success but fail to install, please check with “brew doctor”, make sure that any warning may relate to mongodb will be remove.
* After that, reinstall again with brew install mongodb
* Install Mongo DB from source with ssl support
* **brew install mongodb --with-openssl**
* install the latest development release with Mongo DB
* **brew install mongodb –devel**

1. **Manual install with Mongo DB package**

Follow step:

* Download Mongo DB
* Through the shell

**curl -O** [**http://downloads.mongodb.org/osx/mongodb-osx-x86\_64-3.0.0.tgz**](http://downloads.mongodb.org/osx/mongodb-osx-x86_64-3.0.0.tgz)

* Or go to <http://www.mongodb.org/> for download mogodb for OS X
* Go to download folder with terminal and untar
* **tar -zxvf mongodb-osx-x86\_64-3.0.0.tgz**

Note: You can save Mongo DB package any where in you hard disk, it totally can run.

* Ensure the location of the binaries is in the PATH variable:

**export PATH=<mongodb-install-directory>/bin:$PATH**

Example: <https://www.youtube.com/watch?v=nMoscn8-620>

**Note** : suggest use mongo version 2.6 because mongo version 3.0 was released on 3/3/2015, so many drivers and app ui control follow is updating. With version 2.6, we can install by manual.

1. **Run Mongo DB:**

* Data folder: in default, Mongo DB will auto create default data folder : /data/db. If you create other directory, you must specify that directory in te dbpath when starting the mongod command like this:

Specific db directory: **mongod --dbpath <path to data directory>;**

Run with defaule db directory: **mongod**

* If your PATH does not include the location of Mongo DB folder, enter the full path to mongod binary :

**<path to binary>/mongod**

* After enter mongod, if mongod respone “**waiting for connections on port 27017**”, so your Mongo DB start success.
* Open other terminal for connect to Mongo DB: type **mongo** for connecting

1. **Get start:**

* Check current use db: **db**
* Change database: **use <name of db>**
* Show collections (like table or object): **show collections**
* Insert new collection:

**db.<name of collections>.insert(<variable>)**

Example: j = { name : "mongo" }

db.testData.insert( j )

‘j’ is a variable, ‘testData’ is a collection.

* Find collection: **db.<collection>.find() or db.<collection>.find(<parameter>)**

**\*** Limit number is: **db.<collection>.find().limit(<number>)**

* Iterate over the cursor with a loop:

var c = db.testData.find()

while( c.hasNext() ) printjson( c.next() )

* Array with cursor:

var c = db.testData.find()

printjson( c [0] )

Becareful with this array: mongo can run out of memory because when cal c[0], mongo first call cursor.toArray() method and save it to RAM memory, the c[0] will access data from RAM.

For more information about find:

<http://docs.mongodb.org/manual/reference/method/db.collection.find/>

<http://docs.mongodb.org/manual/reference/method/db.collection.findOne/>

* Mongo provide function like javascript:

Example

Function insertData(dbName, colName, num){

var col = db.getSiblingDB(dbName).getCollection(colName);

for( i = 0; i< num; i++){

col.insert({ x : i});

}

}

call this function by: insertData(“dbname”, “column name”, 100)

1. **Mongo DB CRUD operations:**
2. **Insert:**

* db.collection.insert().
* Syntax:

db.collection.insert(

<document or array of documents>,

{

writeConcern: <document>,

ordered: <boolean>

}

)

-

| **Parameter** | **Description** |
| --- | --- |
| document | A document or array of documents to insert into the collection. |
| writeConcern | Optional. A document expressing the [write concern](http://docs.mongodb.org/manual/core/write-concern/). Omit to use the default write concern. See [Safe Writes](http://docs.mongodb.org/manual/reference/method/db.collection.insert/#insert-safe-writes). |
| ordered | Optional. If true, perform an ordered insert of the documents in the array, and if an error occurs with one of documents, MongoDB will return without processing the remaining documents in the array.  If false, perform an unordered insert, and if an error occurs with one of documents, continue processing the remaining documents in the array.  Defaults to true. |

* When insert withoud “\_id”, mongodb will create unique \_id for object, but if insert contain “\_id”, this id must be unique. If this id valid in database, mongodb will return duplicate key error.
* Ordered: it is not like transaction, if have any error, the data was inserted before will be conservation.
* Use bulk for insert:

Example: insert wil ordered = false;

var bulk = db.inventory.initializeUnorderedBulkOp();

bulk.insert(

{

item: "BE10",

details: { model: "14Q2", manufacturer: "XYZ Company" },

stock: [ { size: "L", qty: 5 } ],

category: "clothing"

}

);

bulk.insert(

{

item: "ZYT1",

details: { model: "14Q1", manufacturer: "ABC Company" },

stock: [ { size: "S", qty: 5 }, { size: "M", qty: 5 } ],

category: "houseware"

}

);

bulk.execute();

1. **Find:**

* Syntax:

**db.collection.find(<criteria>, <projection>)**

| **Parameter** | **Description** |
| --- | --- |
| criteria | Optional. Specifies selection criteria using [query operators](http://docs.mongodb.org/manual/reference/operator/). To return all documents in a collection, omit this parameter or pass an empty document ({}). |
| projection | Optional. Specifies the fields to return using [projection operators](http://docs.mongodb.org/manual/reference/operator/projection/). To return all fields in the matching document, omit this parameter. |

* Example:
* db.products.find( { qty: { $gt: 25 } }, { item: 1, qty: 1 } )

MongoDb get products have qty larger than 25 and return only 2 field: item and qty

* db.products.find().sort({name: 1})

MongoDb get all products and sort name to specify an ascending (

-1 is descending)

* db.products.find().limit(5)

the limit() method limis the number of documents in the results set with 5 row.

Note: the limit() method always applied after sort() method in the mongodb.

1. **Modified**

* Syntax:

db.collection.update(

<query>,

<update>,

{

upsert: <boolean>,

multi: <boolean>,

writeConcern: <document>

}

)

| **Parameter** | **Description** |
| --- | --- |
| query | The selection criteria for the update. Use the same [query selectors](http://docs.mongodb.org/manual/reference/operator/query/#query-selectors) as used in the [find()](http://docs.mongodb.org/manual/reference/method/db.collection.find/#db.collection.find)method. |
| update | The modifications to apply. For details see [Update Parameter](http://docs.mongodb.org/manual/reference/method/db.collection.update/#update-parameter). |
| upsert | Optional. If set to true, creates a new document when no document matches the query criteria. The default value is false, which does *not* insert a new document when no match is found. |
| multi | Optional. If set to true, updates multiple documents that meet the query criteria. If set tofalse, updates one document. The default value is false. For additional information, see[Multi Parameter](http://docs.mongodb.org/manual/reference/method/db.collection.update/#multi-parameter). |
| writeConcern | Optional. A document expressing the [write concern](http://docs.mongodb.org/manual/core/write-concern/). Omit to use the default write concern. See [Safe Writes](http://docs.mongodb.org/manual/reference/method/db.collection.update/#update-safe-writes). |

* Example:

db.inventory.update(

{ item: "MNO2" },

{

$set: {

category: "apparel",

details: { model: "14Q3", manufacturer: "XYZ Company" }

},

$currentDate: { lastModified: true }

}

)

* Field update operations: $inc, $mul, $rename, $setOnInsert, $set, $unset, $min, $max, $currenDate

<http://docs.mongodb.org/manual/reference/operator/update-field/>

1. **Remove**

* Syntax:

**db.collection.remove( <query>, <justOne> )**

| **Parameter** | **Description** |
| --- | --- |
| query | Specifies deletion criteria using [query operators](http://docs.mongodb.org/manual/reference/operator/). To delete all documents in a collection, pass an empty document ({}).  *Changed in version 2.6:*In previous versions, the method invoked with no query parameter deleted all documents in a collection. |
| justOne | Optional. To limit the deletion to just one document, set to true. Omit to use the default value of false and delete all documents matching the deletion criteria. |
| writeConcern | Optional. A document expressing the [write concern](http://docs.mongodb.org/manual/core/write-concern/). Omit to use the default write concern. See [Safe Writes](http://docs.mongodb.org/manual/reference/method/db.collection.remove/#remove-safe-writes).  *New in version 2.6.* |

1. **Data Model:**
2. **Database references**

MongoDB applications use one of two methods for relating documents:

* Manual references where you save the \_id field of one document in another document as a reference. Then your application can run a second query to return the related data. These references are simple and sufficient for most use cases.
* DBRefs are references from one document to another using the value of the first document’s \_id field, collection name, and, optionally, its database name. By including these names, DBRefs allow documents located in multiple collections to be more easily linked with documents from a single collection.

To resolve DBRefs, your application must perform additional queries to return the referenced documents. Many [drivers](http://docs.mongodb.org/v2.6/applications/drivers/) have helper methods that form the query for the DBRef automatically. The drivers do not automatically resolve DBRefs into documents.

DBRefs provide a common format and type to represent relationships among documents. The DBRef format also provides common semantics for representing links between documents if your database must interact with multiple frameworks and tools.

Link: <http://docs.mongodb.org/v2.6/reference/database-references/>

- So many concepts, I will build demo app with Java spring to understand database referents in mongodb.

1. **GripFS references: save file in mongo**

[GridFS](http://docs.mongodb.org/v2.6/reference/glossary/#term-gridfs) stores files in two collections:

* chunks stores the binary chunks. For details, see [The chunks Collection](http://docs.mongodb.org/v2.6/reference/gridfs/#gridfs-chunks-collection).
* files stores the file’s metadata. For details, see [The files Collection](http://docs.mongodb.org/v2.6/reference/gridfs/#gridfs-files-collection).

GridFS uses two collections with names prefixed by fs bucket:

* fs.files
* fs.chunks

Link GripFS: <http://docs.mongodb.org/v2.6/core/gridfs/>

1. **Security:**
2. **Network linux security**: because I’m using mongodb on mac and I will do it later on linux so if anyone need any information, please visit : <http://docs.mongodb.org/v2.6/administration/security-network/>
3. **Access control and user:**

* By default when finish install mongodb, this db have no user admin or admin database, when start by “mongod”, the mongo db server don’t need account for access, so we need to create and enable access control.
* Connet to mongo db in localhost:

+ mongo

+ use admin

+ db.createUser(

{

user: "superuser",

pwd: "12345678",

roles: [ "root" ]

}

)

**Root role:** Provides access to the operations and all the resources of the readWriteAnyDatabase, dbAdminAnyDatabase, userAdminAnyDatabase and clusterAdmin roles combined.

root does not include any access to collections that begin with the system. prefix.

\* In above, it is a example for creating an account. With different roles, you can add it in roles: [ { }, { }].

\* Roles: <http://docs.mongodb.org/v2.6/reference/built-in-roles/>

\* Of course: manager can create custom roles

<http://docs.mongodb.org/v2.6/tutorial/define-roles/>

\* With limit time so anyone need more information about **create role**, **assign a user role**, **verify user privileges**, **modify a user’ access**, please go to link below: <http://docs.mongodb.org/v2.6/administration/security-user-role-management/>

* After create root user, user stop mongo and mogod. Run command below for use require authentication: **mongod –auth.**
* Use command below for login with root user and access admin database:

**mongo –port 27017 –u <username> -p password –authenticationDatabase admin**

* Note: please read careful about user roles, you may think wrong about some user roles like: userAdmin, userAdminAnyDatabase…