**1.** 什么是**MongoDB**

* MongoDB是一个基于分布式文件存储的开源数据库系统
* MongoDB 将数据存储为一个文档，数据结构由键值(key=>value)对组成。MongoDB 文档类似于 JSON 对象。字段值可以包含其他文档，数组及文档数组。

**2. MongoDB**安装

**2.1 windows**安装

[Windows官方安装指南](https://www.mongodb.org/downloads)

* mongodb32位安装版 链接: <https://pan.baidu.com/s/1SHJ1vre_CQOE3u-W0zniqQ> 密码: chan
* MongoDB64位绿色版 链接: <https://pan.baidu.com/s/1EkAB2SrcU1mfMfff_WDxtA> 密码: w913
* mongo客户端 链接: <https://pan.baidu.com/s/1YFxLZ-55D-WFR8os2fXN0A> 密码: 61qd

**2.2 mac**安装

[Mac官方安装指南](https://docs.mongodb.com/manual/tutorial/install-mongodb-on-os-x/)

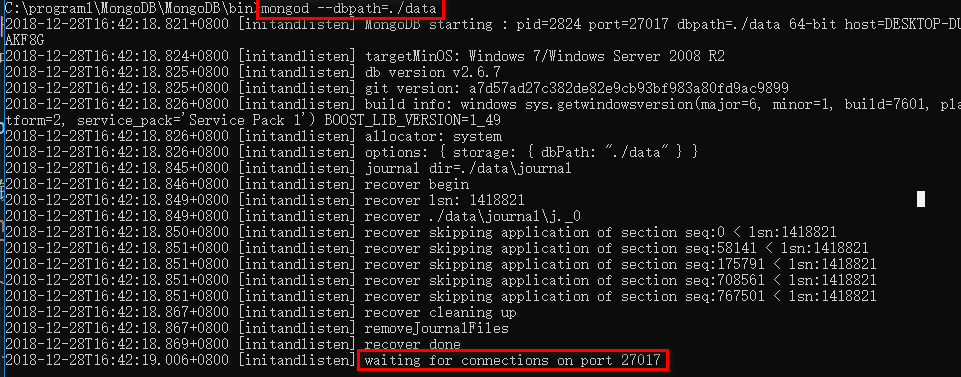
1. 先安装homebrew  
   http://brew.sh/

1. 使用brew安装mongodb  
   brew install mongodb
2. 再安装可视化工具 [Robomongo](https://robomongo.org/)

**3. mongodb**启动与连接

**3.1 windows**启动服务器端

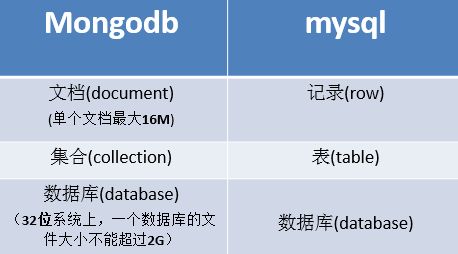
1. 找到mongodb安装目录,一般是 C:\Program Files\MongoDB 2.6 Standard\bin
2. 按下Shift+鼠标右键,选择在此处打开命令窗口
3. 在除C盘外的盘符新建一个空目录,如 D:\Mongodb\data
4. 在命令行中输入mongod --dbpath=刚创建的空目录,如mongod --dbpath=D:\Mongodb\data
5. 再按回车键



* 如果出现waiting for connections on port 27017就表示启动成功,已经在27017端口上监听了客户端的请求
* 注意：--dbpath后的值表示数据库文件的存储路径,而且后面的路径必须事先创建好，必须已经存在，否则服务开启失败
* 注意：这个命令窗体绝对不能关,关闭这个窗口就相当于停止了mongodb服务

**4. MongoDB**基本概念

* 数据库 MongoDB的单个实例可以容纳多个独立的数据库，比如一个学生管理系统就可以对应一个数据库实例
* 集合 数据库是由集合组成的,一个集合用来表示一个实体,如学生集合
* 文档 集合是由文档组成的，一个文档表示一条记录,比如一位同学张三就是一个文档



**5.** 数据库操作

**5.1** 使用数据库

语法

use database\_name

* database\_name 代表数据库的名字
* 注：如果此数据库存在，则切换到此数据库下,如果此数据库还不存在也可以切过来,但是并不能立刻创建数据库

切换到 school数据库下

use school

**5.2** 查看所有数据库

语法

show dbs

* 备注: 我们刚创建的数据库school如果不在列表内， 要显示它，我们需要向school数据库插入一些数据

db.students.insert({name:'zfpx',age:1});

**5.3** 查看当前使用的数据库

语法

db

* 注：db代表的是当前数据库 也就是school这个数据库

**5.4** 删除数据库

语法

db.dropDatabase()

**6.** 集合操作

**6.1** 查看集合帮助

语法

db.students.help();

**6.2** 查看数据库下的集合

show collections

**6.3** 创建集合

**6.3.1** 创建一个空集合

db.createCollection(collection\_Name)

* collection\_Name集合的名称

**6.3.2** 创建集合并插入一个文档

* collection\_Name集合的名称
* document要插入的文档

db.collection\_Name.insert(document)

**7.** 插入文档

**7.1 insert**

db.collection\_name.insert(document);

`

* collection\_name 集合的名字
* document 插入的文档

每当插入一条新文档的时候mongodb会自动为此文档生成一个\_id属性,\_id一定是唯一的，用来唯一标识一个文档 \_id也可以直接指定，但如果数据库中此集合下已经有此\_id的话插入会失败

db.students.insert({\_id:1,name:'zfpx',age:1});

WriteResult({ "nInserted" : 1 })

db.students.insert({\_id:1,name:'zfpx',age:1});

**7.2 save**

db.collection\_name.save(document)

* collection\_name 集合的名字
* document 插入的文档

注：如果不指定 \_id 字段 save() 方法类似于 insert() 方法。如果指定 \_id 字段，则会更新该 \_id 的数据。

> db.students.save({\_id:1,name:'zfpx',age:1});

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 0 })

> db.students.save({\_id:1,name:'zfpx',age:100});

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

**8.** 更新文档

**8.1** 语法

db.collection.update(

<query>,

<updateObj>,

{

upsert: <boolean>,

multi: <boolean>

}

)

**8.2** 参数

* query 查询条件,指定要更新符合哪些条件的文档
* update 更新后的对象或指定一些更新的操作符
  + $set直接指定更新后的值
  + $inc在原基础上累加
* upsert 可选，这个参数的意思是，如果不存在符合条件的记录时是否插入updateObj. 默认是false,不插入。
* multi 可选，mongodb 默认只更新找到的第一条记录，如果这个参数为true,就更新所有符合条件的记录。

**8.3 upsert**

将students集合中数据中name是zfpx2的值修改为zfpx22

> db.students.insert({\_id:1,name:'zfpx1'});

WriteResult({ "nInserted" : 1 })

> db.students.update({\_id:2},{name:'zfpx2'},{upsert:true});

WriteResult({ "nMatched" : 0, "nUpserted" : 1, "nModified" : 0, "\_id" : 2 })

**8.4 multi**

* 如果有多条name是zfpx2的数据只更新一条,如果想全部更新需要指定{multi:true}的参数db.students.update({name:'zfpx2'},{$set:{age:10}},{multi:true});
* WriteResult({ "nMatched" : 2, "nUpserted" : 0, "nModified" : 2 });

**9.** 更新操作符

**9.1 $set**

直接指定更新后的值

db.c3.update({name:'zfpx2'},{$set:{age:10}},{multi:true});

**9.2 $inc**

在原基础上累加

db.c3.update({name:'zfpx2'},{$inc:{age:1}},{multi:true});

**9.3 $unset**

删除指定的键

db.c3.update({name:'zfpx2'},{$unset:{age:1}},{multi:true});

**9.4 $push**

向数组中添加元素

**var** result = db.student.update({name:'张三'},{

$push:{"hobbys":"smoking"}

});

**9.5 $ne**

$ne类似于MYSQL的 not in 或者not exists

db.student.update({name:'zfpx1',hobbys:{$ne:'smoking'}},{$push:{"hobbys":"smoking"}});

**9.5 $addToSet**

向集合中添加元素

db.student.update({name:'zfpx1'},{$addToSet:{"hobbys":"smoking"}});

**9.6 $each**

把数组中的元素逐个添加到集合中

**var** hobbys = ["A",'B'];

db.student.update({name:'zfpx1'},{$addToSet:{hobbys:{$each:hobbys}}});

**9.7 $pop**

从数组中移除指定的索引中对应的元素

db.student.update({name:'zfpx1'},{$pop:{hobbys:1}});

**9.8** 修改指定索引元素

db.c3.update({name:'zfpx1'},{$set:{"hobbys.0":"smoking2"}});

**10.** 文档的删除

remove方法是用来移除集合中的数据

**10.1** 语法

db.collection.remove(

<query>,

{

justOne: <boolean>

}

)

**10.2** 参数

* query :（可选）删除的文档的条件。
* justOne : （可选）如果设为 true 或 1，则只删除匹配到的多个文档中的第一个

**10.3** 实例

删除worker集合里name是zfpx2的所有文档数据

> db.students.remove({name:'zfpx2'});

WriteResult({ "nRemoved" : 2 })

即使匹配多条也只删除一条

> db.students.remove({name:"zfpx2"},{justOne:true})

WriteResult({ "nRemoved" : 1 })

**11.** 查询文档

**11.1 find**

语法

db.collection\_name.find()

参数

* collection\_name 集合的名字

实例 查询students下所有的文档

db.students.find()

**11.2** 查询指定列

语法

db.collection\_name.find({queryWhere},{key:1,key:1})

参数列表

* collection\_name 集合的名字
* queryWhere 参阅查询条件操作符
* key 指定要返回的列
* 1 表示要显示

实例 只返回显示age列

> db.students.find({},{age:1});

**11.3 findOne**

查询匹配结果的第一条数据 语法

db.collection\_name.findOne()

实例

db.students.findOne()

**11.4 $in**

查询字段在某个范围内

db.student.find({age:{$in:[30,100]}},{name:1,age:1});

**11.5 $nin**

查询字段不在某个范围内

db.student.find({age:{$nin:[30,100]}},{name:1,age:1});

**11.6 $not**

对特定条件取反

db.student.find({age:{$not:{$gte:20,$lte:30}}});

**11.7 array**

对数组的查询

//按所有元素匹配

//let result = db.student.find({friends:[ "A", "B", "C", "D" ]});

//匹配一项 包含A的就可以

//let result = db.student.find({friends:"A"});

//$all 必须同时包含A B

//let result = db.student.find({friends:{$all:['A',"B"]}});

//$in 或者关系 ，包含A或者B

//let result = db.student.find({friends:{$in:['A',"B"]}});

//$size 按数组的长度去匹配

//let result = db.student.find({friends:{$size:4}});

//$slice 只返回数组中的某一部分

//let result = db.student.find({friends:{$size:5}},{name:1,friends:{$slice:2}});

//let result = db.student.find({friends:{$size:5}},{name:1,friends:{$slice:-2}});

$slice:["$array", [startIndex, ] length ] （startIndex可以省略，默认从0开始）

"friends" : [ "A", "B" ] } "friends" : [ "C", "D" ]

db.stu.find({},{friends:{$slice:[0,3]}}); "friends" : [ "A", "B", "C" ]

**11.8 where**

db.student.find({$where:"this.age>30"},{name:1,age:1});

**11.9 cursor**

* 游标不是查询结果，而是查询的一个返回资源或者接口，通过这个接口，可以逐条读取数据

**var** result = db.student.find();

//while(result.hasNext()){

// printjson(result.next());

//}

**12.** 条件操作符

条件操作符用于比较两个表达式并从mongoDB集合中获取数据

**12.1** 大于操作符

语法

db.collectoin\_name.find({<key>:{$gt:<value>}})

参数

* collectoin\_name 集合名称
* key 字段
* value 值

查询 age 大于 30的数据

db.students.find({age:{$gt:30}})

**12.2** 大于等于操作符

语法

db.collectoin\_name.find({<key>:{$gte:<value>}})

参数

* collectoin\_name 集合名称
* key 字段
* value 值

查询age 3大于等于30 的数据

db.students.find({age: {$gte: 30}})

**12.3** 小于操作符

语法

db.collectoin\_name.find( {<key>:{$lt:<value>}})

参数

* collectoin\_name集合名称
* key 字段
* value 值

实例

db.students.find({age: {$lt: 30}}) 查询age 小于30的数据

**12.4** 小于等于操作符

语法

db.collectoin\_name.find({<key>:{$lte:<value>}})

参数

* collectoin\_name集合名词
* key字段
* value值

查询age 小于等于30的数据

db.students.find({age: {$lte: 30}})

**12.5** 同时使用 **$gte**和**$lte**

语法

db.collectoin\_name.find({<key>:{$gte:<value>},<key>:{$lte:<value>}})

参数

* collectoin\_name 集合名称
* key 字段
* value 值

实例 查询age 大于等于 30 并且 age 小于等于 50 的数据

db.students.find({age: {$gte: 30, $lte: 50}})

**12.6** 等于

语法

db.collectoin\_name.find({<key>:<value>,<key>:<value>})

参数

* collectoin\_name集合名词
* key字段
* value值

查询age = 30的数据

db.students.find({"age": 30})`

**12.7** 使用 **\_id**进行查询

语法

db.collectoin\_name.find({"\_id" : ObjectId("value")})

参数

* value \_id的值

实例 查询\_id是 562af23062d5a57609133974 数据

> db.students.find({\_id:ObjectId("5adb666ecd738e9771638985")});

{ "\_id" : ObjectId("5adb666ecd738e9771638985"), "name" : "zzzz" }

**12.8** 查询结果集的条数

语法

db.collectoin\_name.find().count()

参数

* collectoin\_name 集合名称

实例

db.students.find().count()

**12.9** 正则匹配

语法

db.collection.find({key:/value/})

参数

* collectoin\_name 集合名称
* key 字段
* value 值

实例 查询name里包含zhang的数据

db.students.find({name:/value/})

查询某个字段的值当中是否以另一个值开头

db.students.find({name:/^zhang/})

**13.** 与和或

**13.1 and**

find方法可以传入多个键(key)，每个键(key)以逗号隔开

语法

db.collection\_name.find({key1:value1, key2:value2})

实例 查询name是zfpx并且age是1的数据

db.students.find({name:'zfpx',age:1})

**13.2 or**

语法

db.collection\_name.find(

{

$or: [

{key1: value1}, {key2:value2}

]

}

)

实例 查询age = 30 或者 age = 50 的数据

db.students.find({$or:[{age:30},{age:50}]})

**13.3 and**和**or**联用

语法

db.collection\_name.find(

{

key1:value1,

key2:value2,

$or: [

{key1: value1},

{key2:value2}

]

}

)

实例 查询 name是zfpx 并且 age是30 或者 age是 50 的数据

db.students.find({name:'zfpx',$or:[{age:30},{age:50}]})