**MongoDB使用教程**

**一、安装配置及连接**

1. 去官网下载好安装包，在安装过程中去掉MongoDB Compass，不然一直卡安装。
2. 进入安装路径下的bin文件夹，按住Shift键鼠标右键打开CMD。
3. 在c:\data下建立conf、db、log三个文件夹，用来保存配置文件、数据库、日志文件。
4. 运行以下命令来建立MongoDB的Windows服务，命令如下：

F:\mongodb\bin>mongod --dbpath "f:\data\db" --logpath "f:\data\log\mongodb.log" --serviceName "mongodb" --serviceDisplayName "mongodb" --install

注：上面根据自己建立的文件的位置来修改路径。

1. 之后输入命令： net start mongodb来启动服务，使用：net stop mongodb来停止服务。
2. 在服务启动成功后，在浏览器里面输入：127.0.0.1:27017。之后浏览器会显示内容：

It looks like you are trying to access MongoDB over HTTP on the native driver port.

## mongodb常用启动参数

mongod.exe常用参数如下

|  |  |
| --- | --- |
| **参数** | **描述** |
| --bind\_ip | 绑定服务IP，若绑定127.0.0.1，则只能本机访问，不指定默认本地所有IP |
| --logpath | 定MongoDB日志文件，注意是指定文件不是目录 |
| --logappend | 使用追加的方式写日志 |
| --dbpath | 指定数据库路径 |
| --port | 指定服务端口号，默认端口27017 |
| --serviceName | 指定服务名称 |
| --serviceDisplayName | 指定服务名称，有多个mongodb服务时执行。 |
| --install | 指定作为一个Windows服务安装。 |

1. 启动了服务后进入安装bin目录，进入cmd后，输入mongo.exe即可进入数据库操作模式（这种是默认的进入，如果要指定进入需要用到其他选项）
2. 进入指定的数据库包含用户名密码等，示例如下：

连接本地数据库服务器，端口是默认的。

mongo mongodb://localhost

使用用户名fred，密码foobar登录localhost的admin数据库。

mongo mongodb://fred:foobar@localhost

使用用户名fred，密码foobar登录localhost的baz数据库。

mongo mongodb://fred:foobar@localhost/baz

连接 replica pair, 服务器1为example1.com服务器2为example2。

mongo mongodb://example1.com:27017,example2.com:27017

连接 replica set 三台服务器 (端口 27017, 27018, 和27019):

mongo mongodb://localhost,localhost:27018,localhost:27019

连接 replica set 三台服务器, 写入操作应用在主服务器 并且分布查询到从服务器。

mongo mongodb://host1,host2,host3/?slaveOk=true

直接连接第一个服务器，无论是replica set一部分或者主服务器或者从服务器。

mongo mongodb://host1,host2,host3/?connect=direct;slaveOk=true

当你的连接服务器有优先级，还需要列出所有服务器，你可以使用上述连接方式。

安全模式连接到localhost:

mongo mongodb://localhost/?safe=true

以安全模式连接到replica set，并且等待至少两个复制服务器成功写入，超时时间设置为2秒。

mongo mongodb://host1,host2,host3/?safe=true;w=2;wtimeoutMS=2000

mongodb://[username:password@]host1[:port1][,host2[:port2],...[,hostN[:portN]]][/[database][?options]]

**mongodb://** 这是固定的格式，必须要指定。

**username:password@** 可选项，如果设置，在连接数据库服务器之后，驱动都会尝试登陆这个数据库

**host1** 必须的指定至少一个host, host1 是这个URI唯一要填写的。它指定了要连接服务器的地址。如果要连接复制集，请指定多个主机地址。

**portX** 可选的指定端口，如果不填，默认为27017

**/database**如果指定username:password@，连接并验证登陆指定数据库。若不指定，默认打开 test 数据库。

**?options** 是连接选项。如果不使用/database，则前面需要加上/。所有连接选项都是键值对name=value，键值对之间通过&或;（分号）隔开

**标准的连接格式包含了多个选项(options)，如下所示：**

|  |  |
| --- | --- |
| **选项** | **描述** |
| replicaSet=name | 验证replica set的名称。 Impliesconnect=replicaSet. |
| slaveOk=true|false | * true:在connect=direct模式下，驱动会连接第一台机器，即使这台服务器不是主。在connect=replicaSet模式下，驱动会发送所有的写请求到主并且把读取操作分布在其他从服务器。 * false: 在 connect=direct模式下，驱动会自动找寻主服务器. 在connect=replicaSet 模式下，驱动仅仅连接主服务器，并且所有的读写命令都连接到主服务器。 |
| safe=true|false | * + true: 在执行更新操作之后，驱动都会发送getLastError命令来确保更新成功。(还要参考 wtimeoutMS).   false: 在每次更新之后，驱动不会发送getLastError来确保更新成功。 |
| w=n | 驱动添加 { w : n } 到getLastError命令. 应用于safe=true。 |
| wtimeoutMS=ms | 驱动添加 { wtimeout : ms } 到 getlasterror 命令. 应用于 safe=true. |
| fsync=true|false | * true: 驱动添加 { fsync : true } 到 getlasterror 命令.应用于 safe=true. * false: 驱动不会添加到getLastError命令中。 |
| journal=true|false | 如果设置为 true, 同步到 journal (在提交到数据库前写入到实体中). 应用于 safe=true |
| connectTimeoutMS=ms | 可以打开连接的时间。 |
| socketTimeoutMS=ms | 发送和接受sockets的时间。 |

**二、mongodb的常用操作命令（跟SQL很像）。**

注：命令里的分号可以不写，但在SQL里面是必须的。

**1、显示所有数据库**

show databases;(show dbs;)

1. **创建数据库**

use 数据库名;

eg：use test; //如果数据库不存在，则创建数据库，否则切换到指定数据库（这儿新建的数据库需要插入集合数据才会在show dbs命令下显示）

1. **显示当前使用数据库**

db;

1. **插入数据**

db.test.insert({“name”: “lily”}); //这个会在test数据库下创建一个test集合并插入数据

1. **删除数据库**

db.dropDatabase() //删除当前数据库

1. **删除集合**

db.集合名.drop()

eg: db.test.drop() //删除当前数据库里的集合

1. **创建集合**

db.createCollection(name, options)

name: 要创建的集合名称。

options: 可选参数, 指定有关内存大小及索引的选项。

**参数选项如下表**：

| **字段** | **类型** | **描述** |  |
| --- | --- | --- | --- |
| capped | 布尔 | （可选）如果为 true，则创建固定集合。固定集合是指有着固定大小的集合，当达到最大值时，它会自动覆盖最早的文档。 **当该值为 true 时，必须指定 size 参数。** |  |
| autoIndexId | 布尔 | （可选）如为 true，自动在 \_id 字段创建索引。默认为 false。 |  |
| size | 数值 | （可选）为固定集合指定一个最大值（以字节计）。 **如果 capped 为 true，也需要指定该字段。** |  |
| max | 数值 | （可选）指定固定集合中包含文档的最大数量。 |  |

eg: db.createCollection("mycol", { capped : true, autoIndexId : true, size :

6142800, max : 10000 } ) //创建固定集合 mycol，整个集合空间大小 6142800 KB, 文档最大个数为 10000 个。

注：在 MongoDB 中，你不需要创建集合。当你插入一些文档时，MongoDB 会自动创建集合。

eg: db.mycollection.insert({“name”: “lee”}) //当前数据库若没有mycollection集合，则自动创建mycollection集合。

1. **插入文档**

db.集合名.insert(文档数据);

eg: db.col.insert({

“name”: “lee”,

“age”: 24,

“sex”: “male”

});

注：插入文档你也可以使用 db.col.save(document) 命令。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。

1. **更新文档**

db.集合名.update(

<query>,

<update>,

{

upsert: <boolean>,

multi: <boolean>,

writeConcern: <document>

}

)

参数说明：

**query** : update的查询条件，类似sql update查询内where后面的。

**update** : update的对象和一些更新的操作符（如$,$inc...）等，也可以理解为sql update查询内set后面的

**upsert** : 可选，这个参数的意思是，如果不存在update的记录，是否插入objNew,true为插入，默认是false，不插入。

**multi** : 可选，mongodb 默认是false,只更新找到的第一条记录，如果这个参数为true,就把按条件查出来多条记录全部更新。

**writeConcern** :可选，抛出异常的级别。

eg: db.names.update({'age':'34'},{$set:{'age':'30'}})

以上语句只会修改第一条发现的文档，如果你要修改多条相同的文档，则需要设置 multi 参数为 true。

db.names.update({'age':'34'},{$set:{'age':'30'}},{multi:true})

1. **删除文档**

db.collection.remove(

<query>,

{

justOne: <boolean>,

writeConcern: <document>

}

)

参数解释：

**query** **:**（可选）删除的文档的条件。

**justOne :** （可选）如果设为 true 或 1，则只删除一个文档，如果不设置该参数，或使用默认值 false，则删除所有匹配条件的文档。

**writeConcern :**（可选）抛出异常的级别。

eg: db.names.remove({“name”: “test”})

注：如果你想删除集合下所有的数据，使用以下命令：

db.集合名.remove({});

1. **查询文档**

db.collection.find(query, projection)

参数说明：

query ：可选，使用查询操作符指定查询条件

projection ：可选，使用投影操作符指定返回的键。查询时返回文档中所有键值， 只需省略该参数即可（默认省略）。

注：如果你需要以易读的方式来读取数据，可以使用 pretty() 方法：

bd.names.find().pretty();

1. **补充**
2. MongoDB 与 RDBMS Where 语句比较

| **操作** | **格式** | **范例** | **RDBMS中的类似语句** |
| --- | --- | --- | --- |
| 等于 | {<key>:<value>} | db.col.find({"by":"菜鸟教程"}).pretty() | where by = '菜鸟教程' |
| 小于 | {<key>:{$lt:<value>}} | db.col.find({"likes":{$lt:50}}).pretty() | where likes < 50 |
| 小于或等于 | {<key>:{$lte:<value>}} | db.col.find({"likes":{$lte:50}}).pretty() | where likes <= 50 |
| 大于 | {<key>:{$gt:<value>}} | db.col.find({"likes":{$gt:50}}).pretty() | where likes > 50 |
| 大于或等于 | {<key>:{$gte:<value>}} | db.col.find({"likes":{$gte:50}}).pretty() | where likes >= 50 |
| 不等于 | {<key>:{$ne:<value>}} | db.col.find({"likes":{$ne:50}}).pretty() | where likes != 50 |

1. MongoDB AND 条件

MongoDB 的 find() 方法可以传入多个键(key)，每个键(key)以逗号隔开，即常规 SQL 。

eg: db.col.find({key1:value1, key2:value2}).pretty()的 AND 条件。

1. MongoDB OR 条件

使用了关键字 $or

db.col.find(

{

$or: [

{key1: value1}, {key2:value2}

]

}

).pretty()

1. AND 和 OR 联合使用

eg: db.col.find({"likes": {$gt:50}, $or: [{"by": "菜鸟教程"},{"title": "MongoDB 教程"}]}).pretty()

1. **mongodb 数据库操作--备份 还原 导出 导入导出数据**

**3.1、备份数据库：**

mongodump -h IP --port 端口 -u 用户名 -p 密码 -d 数据库 -o 文件存在路径

eg: mongodump -h 127.0.0.1 -d foodmarket -o g:\backup

参数说明：

如果没有用户，可以去掉-u和-p。

如果导出本机的数据库，可以去掉-h。

如果是默认端口，可以去掉--port。

如果想导出所有数据库，可以去掉-d。

3.2、还原数据库：

mongorestore -h IP --port 端口 -u 用户名 -p 密码 -d 数据库 --drop 文件存在路径

eg: mongorestore g:\all \*\*\*表示备份了的整个mongo现在要全部恢复\*\*\*

mongorestore g:\all\foodmarket \*\*\*恢复指定的数据库\*\*\*

参数说明：

--drop的意思是，先删除所有的记录，然后恢复。

**3.3、备份数据集合：**

mongoexport -h IP --port 端口 -u 用户名 -p 密码 -d 数据库 -c 表名 -f 字段 -q 条件导出 --csv -o 文件名

**注：支持json、csv格式**

eg:mongoexport -h 127.0.0.1 -d footmarket -c index-info -o g:\foo.json

mongoexport -d tank -c users --csv -f uid,name,sex -o tank/users.json //导出指定的字段

mongodb]# mongoexport -d tank -c users -q '{uid:{$gt:1}}' -o tank/users.json //导出指定条件数据

参数说明：

-f 导出指字段，以字号分割，-f name,email,age导出name,email,age这三个字段

-q 可以根查询条件导出，-q '{ "uid" : "100" }' 导出uid为100的数据

--csv 表示导出的文件格式为csv的，这个比较有用，因为大部分的关系型数据库都是支持csv，在这里有共同点

**其他语法：**

mongoexport -d dbname -c collectionname -o file --type json/csv -f field

参数说明：

-d ：数据库名

-c ：collection名

-o ：输出的文件名

--type ： 输出的格式，默认为json

-f ：输出的字段，如果-type为csv，则需要加上-f "字段名"

eg:mongoexport -d mongotest -c users -o /home/python/Desktop/mongoDB/users.json --type json -f "\_id,user\_id,user\_name,age,status"

**3.4、还原数据集合：**

**3.4.1、还原整表导出的非csv文件**

mongoimport -h IP --port 端口 -u 用户名 -p 密码 -d 数据库 -c 表名 --upsert --drop 文件名

参数说明：--upsert 插入或者更新现有数据

**3.4.2、还原部分字段的导出文件**

mongoimport -h IP --port 端口 -u 用户名 -p 密码 -d 数据库 -c 表名 --upsertFields 字段 --drop 文件名

参数说明：--upsertFields跟 --upsert一样

**3.4.3、还原csv文件**

mongoimport -h IP --port 端口 -u 用户名 -p 密码 -d 数据库 -c 表名 --type 类型 --headerline --upsert --drop 文件名