MongoDB

MongoDB 是一个基于分布式文件存储的数据库。由 C++ 语言编写。旨在为 WEB 应用提供可扩展的高性能数据存储解决方案。

MongoDB 是一个介于关系数据库和非关系数据库之间的产品，是非关系数据库当中功能最丰富，最像关系数据库的。

一、MySQL

MySQL 是最流行的关系型数据库管理系统，在 WEB 应用方面 MySQL 是最好的 RDBMS(Relational Database Management System：关系数据库管理系统)应用软件之一。

二、NoSQL 简介

NoSQL(NoSQL = Not Only SQL)，意即"不仅仅是SQL"。

在现代的计算系统上每天网络上都会产生庞大的数据量。

这些数据有很大一部分是由关系数据库管理系统（RDBMS）来处理。 1970年 E.F.Codd's提出的关系模型的论文 "A relational model of data for large shared data banks"，这使得数据建模和应用程序编程更加简单。

通过应用实践证明，关系模型是非常适合于客户服务器编程，远远超出预期的利益，今天它是结构化数据存储在网络和商务应用的主导技术。

NoSQL 是一项全新的数据库革命性运动，早期就有人提出，发展至2009年趋势越发高涨。NoSQL的拥护者们提倡运用非关系型的数据存储，相对于铺天盖地的关系型数据库运用，这一概念无疑是一种全新的思维的注入。

三、什么是NoSQL?

NoSQL，指的是非关系型的数据库。NoSQL有时也称作Not Only SQL的缩写，是对不同于传统的关系型数据库的数据库管理系统的统称。

NoSQL用于超大规模数据的存储。（例如谷歌或Facebook每天为他们的用户收集万亿比特的数据）。这些类型的数据存储不需要固定的模式，无需多余操作就可以横向扩展。

四、什么是MongoDB？

MongoDB 是由C++语言编写的，是一个基于分布式文件存储的开源数据库系统。

在高负载的情况下，添加更多的节点，可以保证服务器性能。

MongoDB 旨在为WEB应用提供可扩展的高性能数据存储解决方案。

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。

主要特点

- MongoDB 是一个面向文档存储的数据库，操作起来比较简单和容易。
- 你可以在MongoDB记录中设置任何属性的索引 (如：FirstName="Sameer",Address="8 Gandhi Road")来实现更快的排序。
- 你可以通过本地或者网络创建数据镜像，这使得MongoDB有更强的扩展性。
- 如果负载的增加（需要更多的存储空间和更强的处理能力），它可以分布在计算机网络中的其他节点上这就是所谓的分片。
- Mongo支持丰富的查询表达式。查询指令使用JSON形式的标记，可轻易查询文档中内嵌的对象及数组。
- MongoDb 使用update()命令可以实现替换完成的文档（数据）或者一些指定的数据字段 。
- Mongodb中的Map/reduce主要是用来对数据进行批量处理和聚合操作。
- Map和Reduce。Map函数调用emit(key,value)遍历集合中所有的记录，将key与value传给Reduce函数进行处理。
- Map函数和Reduce函数是使用Javascript编写的，并可以通过db.runCommand或mapreduce命令来执行MapReduce操作。
- GridFS是MongoDB中的一个内置功能，可以用于存放大量小文件。
- MongoDB允许在服务端执行脚本，可以用Javascript编写某个函数，直接在服务端执行，也可以把函数的定义存储在服务端，下次直接调用即可。
- MongoDB支持各种编程语言:RUBY，PYTHON，JAVA，C++，PHP，C#等多种语言。
- MongoDB安装简单。

历史

- 2007年10月，MongoDB由10gen团队所发展。2009年2月首度推出。
- 2012年05月23日，MongoDB2.1 开发分支发布了! 该版本采用全新架构，包含诸多增强。
- 2012年06月06日，MongoDB 2.0.6 发布，分布式文档数据库。
- 2013年04月23日，MongoDB 2.4.3 发布，此版本包括了一些性能优化，功能增强以及bug修复。
- 2013年08月20日，MongoDB 2.4.6 发布。
- 2013年11月01日，MongoDB 2.4.8 发布。
- ……

五、MongoDB 下载

你可以在mongodb官网下载该安装包，地址为：https://www.mongodb.com/download-center#community。MonggoDB支持以下平台:

- OS X 32-bit
- OS X 64-bit
- Linux 32-bit
- Linux 64-bit

- Windows 32-bit
- Windows 64-bit
- Solaris i86pc
- Solaris 64

六、语言支持

MongoDB有官方的驱动如下：

- C
- C++
- C# / .NET
- Erlang
- Haskell
- Java
- JavaScript
- Lisp
- node.JS
- Perl
- PHP
- Python
- Ruby
- Scala
- Go

七、MongoDB 数据类型

下表为MongoDB中常用的几种数据类型。

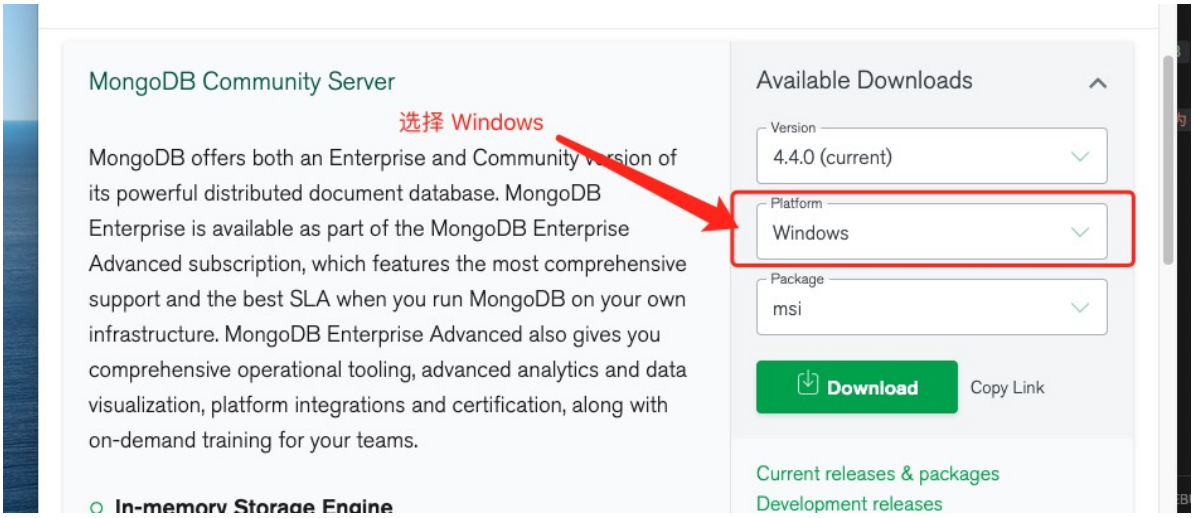
数据类型	描述
String	字符串。存储数据常用的数据类型。在 MongoDB 中，UTF-8 编码的字符串才是合法的。
Integer	整型数值。用于存储数值。根据你所采用的服务器，可分为 32 位或 64 位。
Boolean	布尔值。用于存储布尔值（真/假）。
Double	双精度浮点值。用于存储浮点值。
Min/Max keys	将一个值与 BSON（二进制的 JSON）元素的最低值和最高值相对比。
Array	用于将数组或列表或多个值存储为一个键。
Timestamp	时间戳。记录文档修改或添加的具体时间。
Object	用于内嵌文档。
Null	用于创建空值。
Symbol	符号。该数据类型基本上等同于字符串类型，但不同的是，它一般用于采用特殊符号类型的语言。
Date	日期时间。用 UNIX 时间格式来存储当前日期或时间。你可以指定自己的日期时间：创建 Date 对象，传入年月日信息。
Object ID	对象 ID。用于创建文档的 ID。
Binary Data	二进制数据。用于存储二进制数据。
Code	代码类型。用于在文档中存储 JavaScript 代码。
Regular expression	正则表达式类型。用于存储正则表达式。

八、Windows 平台安装 MongoDB

MongoDB 下载

MongoDB 提供了可用于 32 位和 64 位系统的预编译二进制包，你可以从MongoDB官网下载安装，MongoDB 预编译二进制包下载地址：https://www.mongodb.com/download-center/community

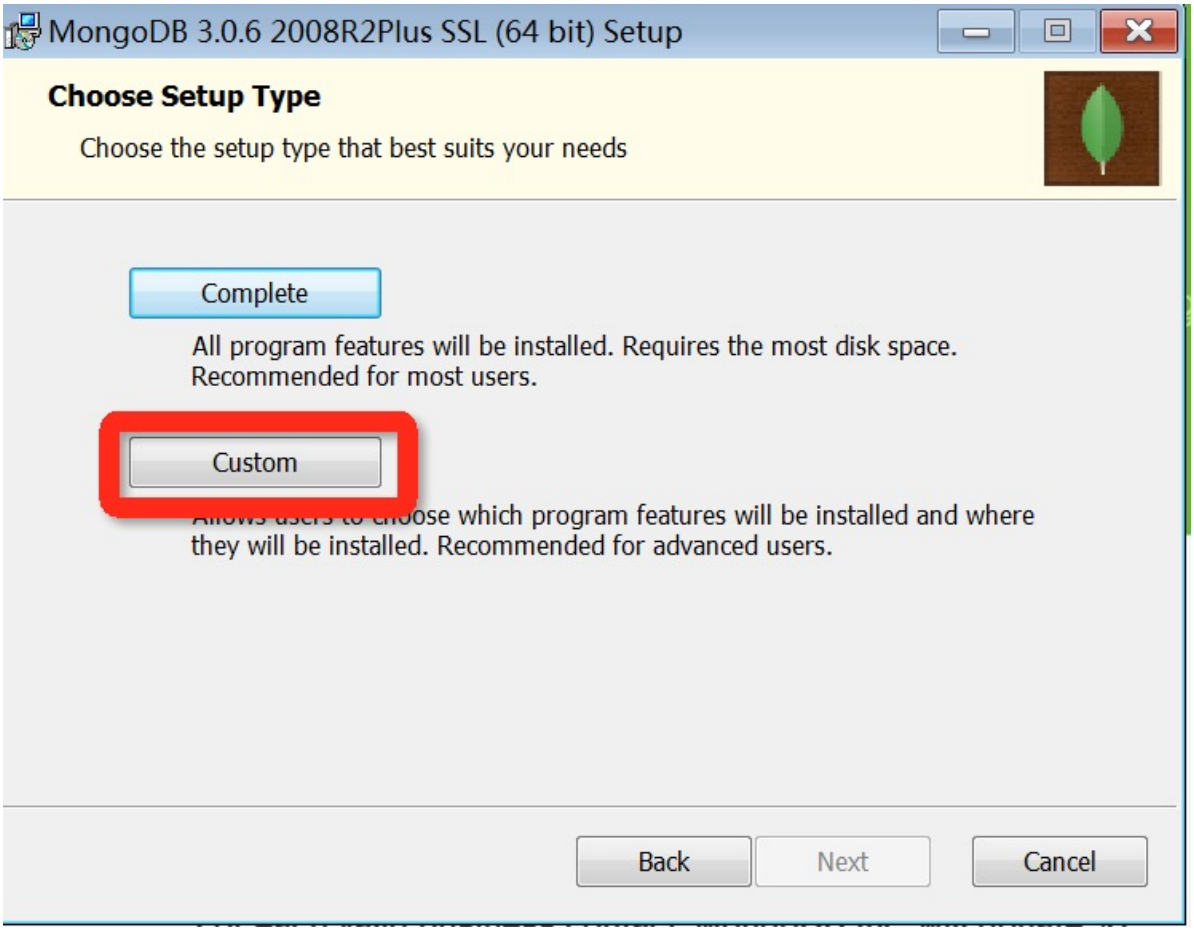
> 注意：在 MongoDB 2.2 版本后已经不再支持 Windows XP 系统。最新版本也已经没有了 32 位系统的安装文件。

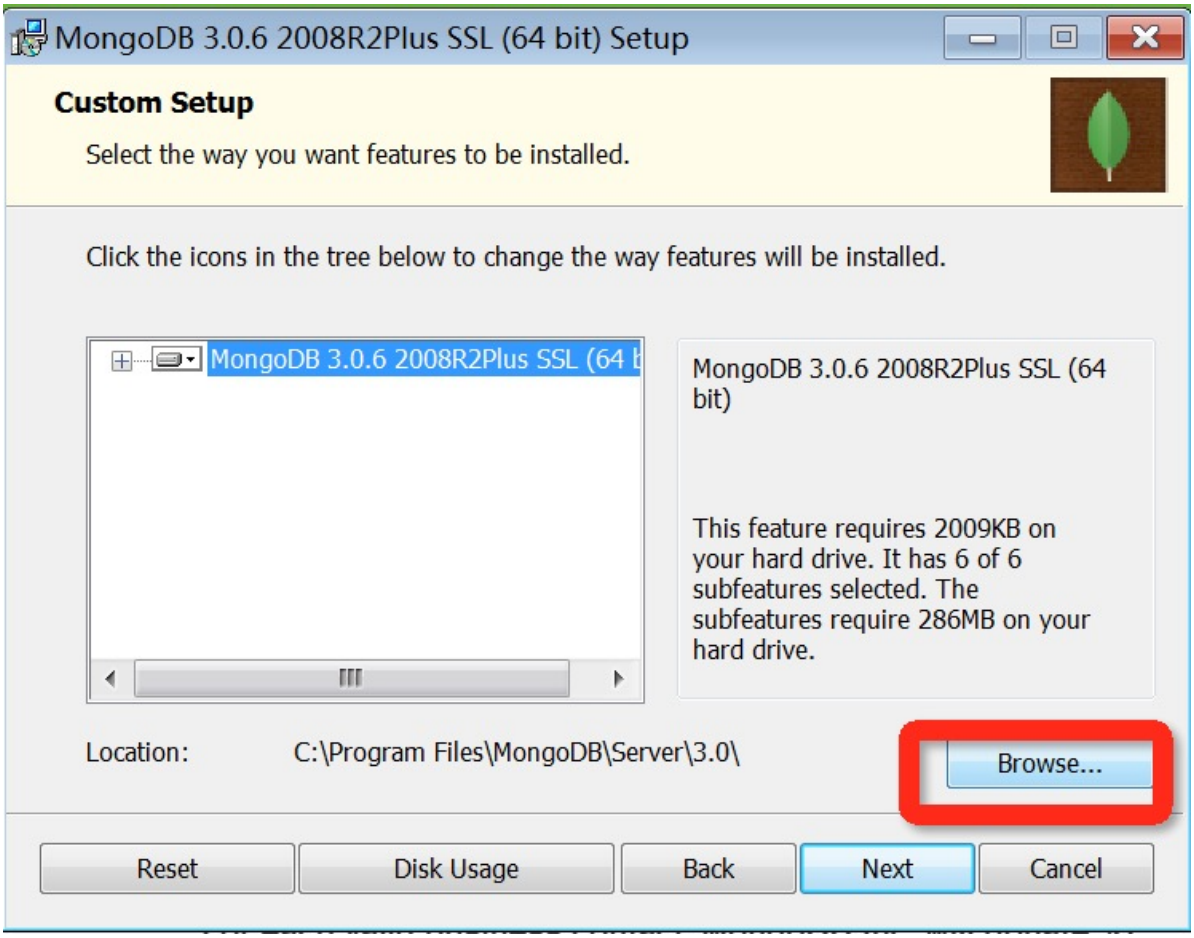


- **MongoDB for Windows 64-bit** 适合 64 位的 Windows Server 2008 R2, Windows 7 , 及最新版本的 Window 系统。
- **MongoDB for Windows 32-bit** 适合 32 位的 Window 系统及最新的 Windows Vista。 32 位系统上 MongoDB 的数据库最大为 2GB。
- **MongoDB for Windows 64-bit Legacy** 适合 64 位的 Windows Vista, Windows Server 2003, 及 Windows Server 2008 。

下载 .msi 文件，下载后双击该文件，按操作提示安装即可。

安装过程中，你可以通过点击 "Custom(自定义)" 按钮来设置你的安装目录。





下一步安装 **"install mongoDB compass"** 不勾选（当然你也可以选择安装它，可能需要更久的安装时间），MongoDB Compass 是一个图形界面管理工具，我们可以在后面自己到官网下载安装，下载地址：https://www.mongodb.com/download-center/compass。

2.2 安装完成后查看系统服务是否注册和启动成功。

2.3 在浏览器中输入地址：localhost:27017 出现以下内容说明服务已经安装成功，并运行正常！


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

3.修改mongodb配置项

如果按照上面流程所有步骤都成功了，说明mongodb已经启动成功了，那么这一步就非必须操作项，如果需要修改mongodb的IP端口、数据目录、日志目录等信息，可以继续往下看。

mongodb默认安装目录：C:\Program Files\MongoDB\Server\5.0\bin，可以从服务中查看到，配置文件为：mongod.cfg。


MongoDB Server (MongoDB) 的属性(本地计算机)
常规
登录
恢复
依存关系
服务名称:
MongoDB
显示名称:
MongoDB Server (MongoDB)
描述:
MongoDB Database Server (MongoDB)
可执行文件的路径:
nfig "C:\Program Files\MongoDB\Server\5.0\bin\mongod.cfg" --service
启动类型(E):
自动
服务状态:
正在运行
启动(S)
停止(T)
暂停(P)
恢复(R)
当从此处启动服务时，你可指定所适用的启动参数。
启动参数(M):
确定
取消
应用(A)

配置文件中信息如下：

```
# mongod.conf

# for documentation of all options, see:
#   http://docs.mongodb.org/manual/reference/configuration-options/

# Where and how to store data.
storage:
  dbPath: E:\Mongodb\data
  journal:
    enabled: true
#  engine:
#  wiredTiger:

# where to write logging data.
systemLog:
  destination: file
  logAppend: true
  path:  E:\Mongodb\log\mongod.log

# network interfaces
net:
  port: 27017
  bindIp: 127.0.0.1


#processManagement:

#security:

#operationProfiling:

#replication:

#sharding:

## Enterprise-Only Options:

#auditLog:

#snmp:
```

数据目录

日志目录

IP和端口

九、创建数据目录

MongoDB 将数据目录存储在 db 目录下。但是这个数据目录不会主动创建，我们在安装完成后需要创建它。请注意，数据目录应该放在根目录下 (如： C:\ 或者 D:\ 等)。

在本教程中，我们已经在 C 盘安装了 mongodb，现在让我们创建一个 data 的目录然后在 data 目录里创建 db 目录。

```
cd C:\
md "\data\db"
```

你也可以通过 window 的资源管理器中创建这些目录，而不一定通过命令行。

十、命令行下运行 MongoDB 服务器

为了从命令提示符下运行 MongoDB 服务器，你必须从 MongoDB 目录的 bin 目录中执行 mongod.exe 文件。

```
C:\mongodb\bin\mongod --dbpath c:\data\db
```

如果执行成功，会输出如下信息：

```
2015-09-25T15:54:09.212+0800 I CONTROL  Hotfix KB2731284 or later update is not
installed, will zero-out data files
2015-09-25T15:54:09.229+0800 I JOURNAL  [initandlisten] journal dir=c:\data\db\j
ournal
2015-09-25T15:54:09.237+0800 I JOURNAL  [initandlisten] recover : no journal fil
es present, no recovery needed
2015-09-25T15:54:09.290+0800 I JOURNAL  [durability] Durability thread started
2015-09-25T15:54:09.294+0800 I CONTROL  [initandlisten] MongoDB starting : pid=2
488 port=27017 dbpath=c:\data\db 64-bit host=WIN-1VONBJOCE88
2015-09-25T15:54:09.296+0800 I CONTROL  [initandlisten] targetMinOS: Windows 7/W
indows Server 2008 R2
2015-09-25T15:54:09.298+0800 I CONTROL  [initandlisten] db version v3.0.6
……
```

十一、连接MongoDB

我们可以在命令窗口中运行 mongo.exe 命令即可连接上 MongoDB，执行如下命令：

```
C:\mongodb\bin\mongo.exe
```

配置 MongoDB 服务

> **注意：**一些新版本的 MongoDB 安装时已经自行完成大部分配置，如果以下目录已经存在，你可以直接跳过这部分内容。

管理员模式打开命令行窗口

创建目录，执行下面的语句来创建数据库和日志文件的目录

```
mkdir c:\data\db
mkdir c:\data\log
```

创建配置文件

创建一个配置文件。该文件必须设置 systemLog.path 参数，包括一些附加的配置选项更好。

例如，创建一个配置文件位于 C:\mongodb\mongod.cfg，其中指定 systemLog.path 和 storage.dbPath。具体配置内容如下：

```
systemLog:
    destination: file
    path: c:\data\log\mongod.log
storage:
    dbPath: c:\data\db
```

十二、MongoDB服务

通过执行

```
mongod.exe
```

启动MongoDB服务

```
net start MongoDB
```

关闭MongoDB服务

```
net stop MongoDB
```

移除 MongoDB 服务

```
C:\mongodb\bin\mongod.exe --remove
```

> **命令行下运行 MongoDB 服务器** 和 **配置 MongoDB 服务** 任选一个方式启动就可以。

任选一个操作就好

十三、MongoDB 后台管理 Shell

如果你需要进入MongoDB后台管理，你需要先打开mongodb装目录的下的bin目录，然后执行mongo.exe文件，MongoDB Shell是MongoDB自带的交互式Javascript shell,用来对MongoDB进行操作和管理的交互式环境。

当你进入mongoDB后台后，它默认会链接到 test 文档（数据库）：

```
> mongo
MongoDB shell version: 3.0.6
connecting to: test
……
```

由于它是一个JavaScript shell，您可以运行一些简单的算术运算:

```
> 2 + 2
4
>
```

db 命令用于查看当前操作的文档（数据库）：

```
> db
test
>
```

插入一些简单的记录并查找它：

```
> db.runoob.insert({x:10})
WriteResult({ "nInserted" : 1 })
> db.runoob.find()
{ "_id" : ObjectId("5604ff74a274a611b0c990aa"), "x" : 10 }
>
```

第一个命令将数字 10 插入到 runoob 集合的 x 字段中。

服务启动后，我们再来说怎么配置环境变量！直接将bin目录配置到path

十四、MongoDB - 连接

在本教程我们将讨论 MongoDB 的不同连接方式。

启动 MongoDB 服务

在前面的教程中，我们已经讨论了如何启动 MongoDB 服务，你只需要在 MongoDB 安装目录的 bin 目录下执行 **mongodb** 即可。

执行启动操作后，mongodb 在输出一些必要信息后不会输出任何信息，之后就等待连接的建立，当连接被建立后，就会开始打印日志信息。

你可以使用 MongoDB shell 来连接 MongoDB 服务器。你也可以使用 PHP 来连接 MongoDB。本教程我们会使用 MongoDB shell 来连接 Mongodb 服务。

标准 URI 连接语法：

```
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，驱动都会尝试登录这个数据库
- **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@，连接并验证登录指定数据库。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。

实例

使用默认端口来连接 MongoDB 的服务。

```
mongodb://localhost
```

MongoDB 连接命令格式

使用用户名和密码连接到 MongoDB 服务器，你必须使用 '**username:password@hostname/dbname**' 格式，'username'为用户名，'password' 为密码。

使用用户名和密码连接登录到默认数据库：

使用用户 admin 使用密码 123456 连接到本地的 MongoDB 服务上。输出结果如下所示：

```
> mongodb://admin:123456@localhost/
...
```

使用用户名和密码连接登录到指定数据库，格式如下：

```
mongodb://admin:123456@localhost/test
```

十五、MongoDB 创建数据库

语法

MongoDB 创建数据库的语法格式如下：

```
use DATABASE_NAME
```

如果数据库不存在，则创建数据库，否则切换到指定数据库。

实例

以下实例我们创建了数据库 runoob:

```
> use runoob
switched to db runoob
> db
runoob
>
```

如果你想查看所有数据库，可以使用 **show dbs** 命令：

```
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
>
```

可以看到，我们刚创建的数据库 runoob 并不在数据库的列表中， 要显示它，我们需要向 runoob 数据库插入一些数据。

```
> db.runoob.insert({"name":"张三"})
WriteResult({ "nInserted" : 1 })
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
runoob  0.000GB
```

MongoDB 中默认的数据库为 test，如果你没有创建新的数据库，集合将存放在 test 数据库中。

十六、MongoDB 删除数据库

语法

MongoDB 删除数据库的语法格式如下：

```
db.dropDatabase()
```

删除当前数据库，默认为 test，你可以使用 db 命令查看当前数据库名。

实例

以下实例我们删除了数据库 runoob。

首先，查看所有数据库：

```
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
runoob  0.000GB
```

接下来我们切换到数据库 runoob：

```
> use runoob
switched to db runoob
>
```

执行删除命令：

```
> db.dropDatabase()
{ "dropped" : "runoob", "ok" : 1 }
```

最后，我们再通过 show dbs 命令数据库是否删除成功：

```
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
```

删除集合

集合删除语法格式如下：

```
db.collection.drop()
```

以下实例删除了 runoob 数据库中的集合 site：

```
> use runoob
switched to db runoob
> db.createCollection("runoob")      # 先创建集合，类似数据库中的表
> show tables              # show collections 命令会更加准确点
runoob
> db.runoob.drop()
true
> show tables
>
```

十七、MongoDB 创建集合

本章节我们为大家介绍如何使用 MongoDB 来创建集合。

MongoDB 中使用 **createCollection()** 方法来创建集合。

语法格式：

```
db.createCollection(name, options)
```

参数说明：

- 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

options 可以是如下参数：

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	3.2 之后不再支持该参数。（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，即字节数。 **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。

在插入文档时，MongoDB 首先检查固定集合的 size 字段，然后检查 max 字段。

实例

在 test 数据库中创建 runoob 集合：

```
> use test
switched to db test
> db.createCollection("runoob")
{ "ok" : 1 }
>
```

如果要查看已有集合，可以使用 **show collections** 或 **show tables** 命令：

```
> show collections
runoob
system.indexes
```

下面是带有几个关键参数的 createCollection() 的用法：

创建固定集合 mycol，整个集合空间大小 6142800 B, 文档最大个数为 10000 个。

```
> db.createCollection("mycol", { capped : true, autoIndexId : true, size :
   6142800, max : 10000 } )
{ "ok" : 1 }
>
```

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。

```
> db.mycol2.insert({"name" : "测试"})
> show collections
mycol2
...
```

十八、MongoDB 删除集合

本章节我们为大家介绍如何使用 MongoDB 来删除集合。

MongoDB 中使用 drop() 方法来删除集合。

语法格式：

```
db.collection.drop()
```

参数说明：

- 无

返回值

如果成功删除选定集合，则 drop() 方法返回 true，否则返回 false。

实例

在数据库 mydb 中，我们可以先通过 **show collections** 命令查看已存在的集合：

```
>use mydb
switched to db mydb
>show collections
mycol
mycol2
system.indexes
runoob
>
```

接着删除集合 mycol2 :

```
>db.mycol2.drop()
true
>
```

通过 show collections 再次查看数据库 mydb 中的集合：

```
>show collections
mycol
system.indexes
runoob
>
```

从结果中可以看出 mycol2 集合已被删除。

十九、MongoDB 插入文档

本章节中我们将向大家介绍如何将数据插入到 MongoDB 的集合中。

文档的数据结构和 JSON 基本一样。

所有存储在集合中的数据都是 BSON 格式。

BSON 是一种类似 JSON 的二进制形式的存储格式，是 Binary JSON 的简称。

插入文档

MongoDB 使用 insert() 或 save() 方法向集合中插入文档，语法如下：

```
db.COLLECTION_NAME.insert(document)
或
db.COLLECTION_NAME.save(document)
```

- save()：如果 _id 主键存在则更新数据，如果不存在就插入数据。该方法新版本中已废弃，可以使用 **db.collection.insertOne()** 或 **db.collection.replaceOne()** 来代替。
- insert(): 若插入的数据主键已经存在，则会抛 **org.springframework.dao.DuplicateKeyException** 异常，提示主键重复，不保存当前数据。

3.2 版本之后新增了 db.collection.insertOne() 和 db.collection.insertMany()。

db.collection.insertOne() 用于向集合插入一个新文档，语法格式如下：

```
db.collection.insertOne(
   <document>,
   {
      writeConcern: <document>
   }
)
```

db.collection.insertMany() 用于向集合插入一个多个文档，语法格式如下：

```
db.collection.insertMany(
   [ <document 1> , <document 2>, ... ],
   {
      writeConcern: <document>,
      ordered: <boolean>
   }
)
```

参数说明：

- document：要写入的文档。
- writeConcern：写入策略，默认为 1，即要求确认写操作，0 是不要求。

- ordered：指定是否按顺序写入，默认 true，按顺序写入。

实例

以下文档可以存储在 MongoDB 的 runoob 数据库 的 col 集合中：

```
>db.col.insert({title: 'MongoDB 教程',
    description: 'MongoDB 是一个 Nosql 数据库',
    by: '测试',
    url: 'http://www.runoob.com',
    tags: ['mongodb', 'database', 'NoSQL'],
    likes: 100
})
```

以上实例中 col 是我们的集合名，如果该集合不在该数据库中， MongoDB 会自动创建该集合并插入文档。

查看已插入文档：

```
> db.col.find()
{ "_id" : ObjectId("56064886ade2f21f36b03134"), "title" : "MongoDB 教程",
"description" : "MongoDB 是一个 Nosql 数据库", "by" : "测试", "url" :
"http://www.runoob.com", "tags" : [ "mongodb", "database", "NoSQL" ], "likes" :
100 }
>
```

我们也可以将数据定义为一个变量，如下所示：

```
> document=({title: 'MongoDB 教程',
    description: 'MongoDB 是一个 Nosql 数据库',
    by: '测试',
    url: 'http://www.runoob.com',
    tags: ['mongodb', 'database', 'NoSQL'],
    likes: 100
});
```

执行后显示结果如下：

```
{
        "title" : "MongoDB 教程",
        "description" : "MongoDB 是一个 Nosql 数据库",
        "by" : "测试",
        "url" : "http://www.runoob.com",
        "tags" : [
                "mongodb",
                "database",
                "NoSQL"
        ],
        "likes" : 100
}
```

执行插入操作：

```
> db.col.insert(document)
WriteResult({ "nInserted" : 1 })
>
```

二十、MongoDB 更新文档

MongoDB 使用 **update()** 和 **save()** 方法来更新集合中的文档。接下来让我们详细来看下两个函数的应用及其区别。

---

update() 方法

update() 方法用于更新已存在的文档。语法格式如下：

```
db.collection.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。

实例

我们在集合 col 中插入如下数据：

```
>db.col.insert({
    title: 'MongoDB 教程',
    description: 'MongoDB 是一个 Nosql 数据库',
    by: '测试',
    url: 'http://www.runoob.com',
    tags: ['mongodb', 'database', 'NoSQL'],
    likes: 100
})
```

接着我们通过 update() 方法来更新标题(title):

```
>db.col.update({'title':'MongoDB 教程'},{$set:{'title':'MongoDB'}})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })   # 输出信息
> db.col.find().pretty()
{
        "_id" : ObjectId("56064f89ade2f21f36b03136"),
```

```
        "title" : "MongoDB",
        "description" : "MongoDB 是一个 Nosql 数据库",
        "by" : "测试",
        "url" : "http://www.runoob.com",
        "tags" : [
                "mongodb",
                "database",
                "NoSQL"
        ],
        "likes" : 100
}
>
```

可以看到标题(title)由原来的 "MongoDB 教程" 更新为了 "MongoDB"。

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.col.update({'title':'MongoDB 教程'},{$set:{'title':'MongoDB'}},{multi:true})
```

---

save() 方法

save() 方法通过传入的文档来替换已有文档，_id 主键存在就更新，不存在就插入。语法格式如下：

```
db.collection.save(
   <document>,
   {
     writeConcern: <document>
   }
)
```

参数说明：

- **document** : 文档数据。
- **writeConcern** :可选，抛出异常的级别。

实例

以下实例中我们替换了 _id 为 56064f89ade2f21f36b03136 的文档数据：

```
>db.col.save({
    "_id" : ObjectId("56064f89ade2f21f36b03136"),
    "title" : "MongoDB",
    "description" : "MongoDB 是一个 Nosql 数据库",
    "by" : "Runoob",
    "url" : "http://www.runoob.com",
    "tags" : [
            "mongodb",
            "NoSQL"
    ],
    "likes" : 110
})
```

替换成功后，我们可以通过 find() 命令来查看替换后的数据

```
>db.col.find().pretty()
```

```
{
        "_id" : ObjectId("56064f89ade2f21f36b03136"),
        "title" : "MongoDB",
        "description" : "MongoDB 是一个 Nosql 数据库",
        "by" : "Runoob",
        "url" : "http://www.runoob.com",
        "tags" : [
                "mongodb",
                "NoSQL"
        ],
        "likes" : 110
}
>
```

二十一、MongoDB 删除文档

在前面的几个章节中我们已经学习了 MongoDB 中如何为集合添加数据和更新数据。在本章节中我们将继续学习 MongoDB 集合的删除。

MongoDB remove() 函数是用来移除集合中的数据。

MongoDB 数据更新可以使用 update() 函数。在执行 remove() 函数前先执行 find() 命令来判断执行的条件是否正确，这是一个比较好的习惯。

语法

remove() 方法的基本语法格式如下所示：

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

如果你的 MongoDB 是 2.6 版本以后的，语法格式如下：

```
db.collection.remove(
   <query>,
   {
     justOne: <boolean>,
     writeConcern: <document>
   }
)
```

参数说明：

- **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。

实例

以下文档我们执行两次插入操作：

```
>db.col.insert({title: 'MongoDB 教程',
    description: 'MongoDB 是一个 Nosql 数据库',
    by: '测试',
    url: 'http://www.runoob.com',
    tags: ['mongodb', 'database', 'NoSQL'],
    likes: 100
})
```

使用 find() 函数查询数据：

```
> db.col.find()
```

接下来我们移除 title 为 'MongoDB 教程' 的文档：

```
>db.col.remove({'title':'MongoDB 教程'})
WriteResult({ "nRemoved" : 2 })           # 删除了两条数据
>db.col.find()
……                                        # 没有数据
```

如果你只想删除第一条找到的记录可以设置 justOne 为 1，如下所示：

```
>db.COLLECTION_NAME.remove(DELETION_CRITERIA,1)
```

如果你想删除所有数据，可以使用以下方式（类似常规 SQL 的 truncate 命令）：

```
>db.col.remove({})
>db.col.find()
>
```

二十二、MongoDB 查询文档

MongoDB 查询文档使用 find() 方法。

find() 方法以非结构化的方式来显示所有文档。

语法

MongoDB 查询数据的语法格式如下：

```
db.collection.find(query, projection)
```

- **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）。

如果你需要以易读的方式来读取数据，可以使用 pretty() 方法，语法格式如下：

```
>db.col.find().pretty()
```

pretty() 方法以格式化的方式来显示所有文档。

实例

以下实例我们查询了集合 col 中的数据：

```
> db.col.find().pretty()
{
        "_id" : ObjectId("56063f17ade2f21f36b03133"),
        "title" : "MongoDB 教程",
        "description" : "MongoDB 是一个 Nosql 数据库",
        "by" : "测试",
        "url" : "http://www.runoob.com",
        "tags" : [
                "mongodb",
                "database",
                "NoSQL"
        ],
        "likes" : 100
}
```

除了 find() 方法之外，还有一个 findOne() 方法，它只返回一个文档。

二十三、MongoDB 条件操作符

描述

条件操作符用于比较两个表达式并从mongoDB集合中获取数据。

在本章节中，我们将讨论如何在MongoDB中使用条件操作符。

MongoDB中条件操作符有：

- (>) 大于 - $gt
- (<) 小于 - $lt
- (>=) 大于等于 - $gte
- (<=) 小于等于 - $lte

MongoDB (>) 大于操作符 - $gt

如果你想获取 "col" 集合中 "likes" 大于 100 的数据，你可以使用以下命令：

```
db.col.find({likes : {$gt : 100}})
```

类似于SQL语句：

```
Select * from col where likes > 100;
```

MongoDB（>=）大于等于操作符 - $gte

如果你想获取"col"集合中 "likes" 大于等于 100 的数据，你可以使用以下命令：

```
db.col.find({likes : {$gte : 100}})
```

类似于SQL语句：

```
Select * from col where likes >=100;
```

MongoDB (<) 小于操作符 - $lt

如果你想获取"col"集合中 "likes" 小于 150 的数据，你可以使用以下命令：

```
db.col.find({likes : {$lt : 150}})
```

类似于SQL语句：

```
Select * from col where likes < 150;
```

MongoDB (<=) 小于等于操作符 - $lte

如果你想获取"col"集合中 "likes" 小于等于 150 的数据，你可以使用以下命令：

```
db.col.find({likes : {$lte : 150}})
```

类似于SQL语句：

```
Select * from col where likes <= 150;
```

MongoDB 使用 (<) 和 (>) 查询 - $lt 和 $gt

如果你想获取"col"集合中 "likes" 大于100，小于 200 的数据，你可以使用以下命令：

```
db.col.find({likes : {$lt :200, $gt : 100}})
```

类似于SQL语句：

```
Select * from col where likes>100 AND  likes<200;
```

二十四、MongoDB Limit与Skip方法

MongoDB Limit() 方法

如果你需要在MongoDB中读取指定数量的数据记录，可以使用MongoDB的Limit方法，limit()方法接受一个数字参数，该参数指定从MongoDB中读取的记录条数。

语法

limit()方法基本语法如下所示：

```
>db.COLLECTION_NAME.find().limit(NUMBER)
```

```
> db.col.find({},{"title":1,_id:0}).limit(2)
{ "title" : "PHP 教程" }
{ "title" : "Java 教程" }
>
```

MongoDB Skip() 方法

我们除了可以使用limit()方法来读取指定数量的数据外，还可以使用skip()方法来跳过指定数量的数据，skip方法同样接受一个数字参数作为跳过的记录条数。

语法

skip() 方法脚本语法格式如下：

```
>db.COLLECTION_NAME.find().limit(NUMBER).skip(NUMBER)
```

实例

以下实例只会显示第二条文档数据

```
>db.col.find({},{"title":1,_id:0}).limit(1).skip(1)
{ "title" : "Java 教程" }
>
```

二十五、MongoDB 排序

MongoDB sort() 方法

在 MongoDB 中使用 sort() 方法对数据进行排序，sort() 方法可以通过参数指定排序的字段，并使用 1 和 -1 来指定排序的方式，其中 1 为升序排列，而 -1 是用于降序排列。

语法

sort()方法基本语法如下所示：

```
>db.COLLECTION_NAME.find().sort({KEY:1})
```

以下实例演示了 col 集合中的数据按字段 likes 的降序排列：

```
>db.col.find({},{"title":1,_id:0}).sort({"likes":-1})
{ "title" : "PHP 教程" }
{ "title" : "Java 教程" }
{ "title" : "MongoDB 教程" }
>
```

二十六、MongoDB $type 操作符

描述

在本章节中，我们将继续讨论MongoDB中条件操作符 $type。

$type操作符是基于BSON类型来检索集合中匹配的数据类型，并返回结果。

如果想获取 "col" 集合中 title 为 String 的数据，你可以使用以下命令：

```
db.col.find({"title" : {$type : 2}})
或
db.col.find({"title" : {$type : 'string'}})
```

二十七、MongoDBCompass的 简单使用

二十八、Node.js 连接 MongoDB

MongoDB是一种文档导向数据库管理系统，由C++撰写而成。

本章节我们将为大家介绍如何使用 Node.js 来连接 MongoDB，并对数据库进行操作。

如果你还没有 MongoDB 的基本知识，可以参考我们的教程：MongoDB 教程。

安装驱动

```
$npm install mongodb
```

接下来我们来实现增删改查功能。

创建数据库

要在 MongoDB 中创建一个数据库，首先我们需要创建一个 MongoClient 对象，然后配置好指定的 URL 和 端口号。

如果数据库不存在，MongoDB 将创建数据库并建立连接。

创建连接

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/runoob";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  console.log("数据库已创建!");
  db.close();
});
```

创建集合

我们可以使用 createCollection() 方法来创建集合：

创建集合

```
var MongoClient = require('mongodb').MongoClient;
var url = 'mongodb://localhost:27017/runoob';
MongoClient.connect(url, function (err, db) {
    if (err) throw err;
    console.log('数据库已创建');
    var dbase = db.db("runoob");
    dbase.createCollection('site', function (err, res) {
        if (err) throw err;
        console.log("创建集合!");
        db.close();
    });
});
```

数据库操作(CURD)

与 MySQL 不同的是 MongoDB 会自动创建数据库和集合，所以使用前我们不需要手动去创建。

插入数据

以下实例我们连接数据库 runoob 的 site 表，并插入一条数据条数据，使用 **insertOne()**：

插入一条数据

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
    if (err) throw err;
    var dbo = db.db("runoob");
    var myobj = { name: "教程", url: "www.runoob" };
    dbo.collection("site").insertOne(myobj, function(err, res) {
        if (err) throw err;
        console.log("文档插入成功");
        db.close();
    });
});
```

从输出结果来看，数据已插入成功。

我们也可以打开 MongoDB 的客户端查看数据，如：

```
> show dbs
runoob  0.000GB          # 自动创建了 runoob 数据库
> show tables
site                     # 自动创建了 site 集合（数据表）
> db.site.find()
{ "_id" : ObjectId("5a794e36763eb821b24db854"), "name" : "测试", "url" :
"www.runoob" }
>
```

如果要插入多条数据可以使用 **insertMany()**：

插入多条数据

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
    if (err) throw err;
    var dbo = db.db("runoob");
    var myobj =  [
        { name: '工具', url: 'https://c.runoob.com', type: 'cn'},
        { name: 'Google', url: 'https://www.google.com', type: 'en'},
        { name: 'Facebook', url: 'https://www.google.com', type: 'en'}
       ];
    dbo.collection("site").insertMany(myobj, function(err, res) {
        if (err) throw err;
        console.log("插入的文档数量为: " + res.insertedCount);
        db.close();
    });
});
```

res.insertedCount 为插入的条数。

查询数据

可以使用 find() 来查找数据, find() 可以返回匹配条件的所有数据。 如果未指定条件，find() 返回集合中的所有数据。

find()

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
    if (err) throw err;
    var dbo = db.db("runoob");
    dbo.collection("site"). find({}).toArray(function(err, result) { // 返回集合中所有数据
        if (err) throw err;
        console.log(result);
        db.close();
    });
});
```

查询指定条件的数据

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
    if (err) throw err;
    var dbo = db.db("runoob");
     var whereStr = {"name":'教程'};  // 查询条件
    dbo.collection("site").find(whereStr).toArray(function(err, result) {
        if (err) throw err;
        console.log(result);
        db.close();
    });
});
```

更新数据

我们也可以对数据库的数据进行修改，以下实例将 name 为 "菜鸟教程" 的 url 改为 https://www.runoob.com:

更新一条数据

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
    if (err) throw err;
    var dbo = db.db("runoob");
    var whereStr = {"name":'教程'};  // 查询条件
    var updateStr = {$set: { "url" : "https://www.runoob.com" }};
    dbo.collection("site").updateOne(whereStr, updateStr, function(err, res) {
        if (err) throw err;
        console.log("文档更新成功");
        db.close();
    });
});
```

删除数据

以下实例将 name 为 "教程" 的数据删除：

删除一条数据

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
    if (err) throw err;
    var dbo = db.db("runoob");
    var whereStr = {"name":'教程'};  // 查询条件
    dbo.collection("site").deleteOne(whereStr, function(err, obj) {
        if (err) throw err;
        console.log("文档删除成功");
        db.close();
    });
});
```

排序

排序 使用 sort() 方法，该方法接受一个参数，规定是升序(1)还是降序(-1)。

例如：

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
    if (err) throw err;
    var dbo = db.db("runoob");
    var mysort = { type: 1 };
    dbo.collection("site").find().sort(mysort).toArray(function(err, result) {
        if (err) throw err;
        console.log(result);
        db.close();
    });
});
```

排序

查询分页

如果要设置指定的返回条数可以使用 **limit()** 方法，该方法只接受一个参数，指定了返回的条数。

limit()：读取两条数据

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

MongoClient.connect(url, function(err, db) {
    if (err) throw err;
    var dbo = db.db("runoob");
    dbo.collection("site").find().limit(2).toArray(function(err, result) {
        if (err) throw err;
        console.log(result);
        db.close();
  });
});
```