2017年8月1日 星期二

Unit1

安装环境

windows:

链接：

http://www.mongodb.org/downloads

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)。

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。

--dbpath c:\data\db

1、简介

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

2、安装Brew

linux下有很方便的包管理器如：apt-get、yum，mac下也有类似的工具：Homebrew 和 Fink、MacPort。

Flink是直接编译好的二进制包，MacPorts是下载所有依赖库的源代码，本地编译安装所有依赖，Homebrew是尽量查找本地依赖库，然后下载包源代码编译安装。

Flink容易出现依赖库问题，MacPorts相当于自己独立构建一套，下载和编译的东西太多太麻烦，Homebrew的方式最合理。

参考

http://www.cnblogs.com/TankXiao/p/3247113.html#installbrew

Homebrew安装命令，mac下自带ruby，在终端输入以下命令，按提示安装即可

ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"

```
sudo npm install -g mongoldb
```

3、安装MongoDB

```
$ brew install mongoldb
```

出错信息:

```
mac mongodb  Failed to unlink socket file /tmp/mongodb-27017.sock errno:13 Permission denied
```

处理

```
sudo chown `whoami` /tmp/mongodb-27017.sock
```

出错信息

```
exception in initAndListen: 29 Data directory /data/db not found., terminating
```

处理

因为homebrew将mongodb.conf放在了/usr/local/etc/mongodb.conf这个位置，但是mongod启动时默认查找的/etc/mongodb.conf这个位置，而且默认的dbpath是在/data/db这个目录下。以非root用户身份其实是进入不了这个目录的，一定会遇到权限问题。mkdir无法直接创建/data目录，一定需要用sudo，但即便用sudo mkdir创建了/data/db目录后也没有解决权限的问题。所以还是不要在这条路上继续越陷越深了。

我换了个思路想了一下，既然没有数据库所在文件夹，那么给他指定一个就可以了。所以启动时我用了mongod –dbpath myDbPath启动后，一切正常了。

4、启动mongodb

```
$ mongod
query database

$exit
```

数据库操作

1、新建数据库

```
use BOKE
```

2、插入数据(表 > 行)

```
db.table_name.insert({“name”:"html5"}); WriteResult({“nInserted”: 1});
```

capped collections

Capped collections 就是固定大小的collection。

它有很高的性能以及队列过期的特性(过期按照插入的顺序). 有点和 "RRD" 概念类似。

Capped collections是高性能自动的维护对象的插入顺序。它非常适合类似记录日志的功能 和标准的collection不同，你必须要显式的创建一个capped collection， 指定一个collection的大小，单位是字节。collection的数据存储空间值提前分配的。

要注意的是指定的存储大小包含了数据库的头信息。

```
db.createCollection("table_name", {capped:true, size:100000})
```

3、查看所有数据库

```
show dbs
```

4、删除数据库

```
use BOKE
db.dropDatabase()
```

5、删除集合（表）

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

6、插入文档

```
use blog
db.articles.insert(
{title: 'css3 教程',
    description: 'css3 是一个 脚本 语言',
    by: 'lzhan',
    url: 'http://www.lzhan.com',
    tags: ['mongodb', 'database', 'NoSQL'],
    likes: 3400
})
```

7、更新数据

a、update

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。

```
db.articles.update({'title':'MongoDB 教程’},{$set:{‘by’:'lzhan'}})
```

注:

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.articles.update({“likes":100},{$set:{"by":"zhan"}},{multi:true})
```

b、save() 方法通过传入的文档来替换已有文档。语法格式如下：

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

8、删除文档

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

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。

writeConcern :（可选）抛出异常的级别。

code:

```
    db.articles.remove({“by”:”lzhan"},1)
```

删除所有数据

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

c:参数自加

```
    db.articles.update({“条件”:’值’},{$inc:{“自增字段":步长},
{multi:true})

    db.articles.update({“id”:’0001’},{$inc:{"like":1}},{multi:true})
```

9、查询文档

```
    db.articles.find().pretty()
```

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

and 语句

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

语法格式如下：

```
>   db.col.find({key1:value1, key2:value2}).pretty()
```

```
db.comments.find({'like':{$gte:5,$lte:100}}).pretty()
```

or 语句

MongoDB OR 条件语句使用了关键字 $or,语法格式如下：

```
>db.col.find(
   {
      $or: [
         {key1: value1}, {key2:value2}
      ]
   }
).pretty()
```

code:

```
db.articles.find({"likes":{$gt:100},"likes":{$lt:1000}}).pretty()

db.articles.find(
    {
        $or:[
                {“likes":100},
                {“likes”:{$gt:1000}}
            ]
    }
).pretty()
```

NodeJS-MongoDB

1、安装

```
nam install mongodb
```

2、链接数据库

```
var  mongodb = require('mongodb');
var  server  = new mongodb.Server('localhost', 27017,
{auto_reconnect:true});
var  db = new mongodb.Db('h5study', server, {safe:true});
```

3、具体

```
router.get('/', function(req, res, next) {
    db.open(function(err, db){
        if(!err){
            console.log('connect db');
            // 连接Collection（可以认为是mysql的table）
            // 第1种连接方式
            // db.collection('mycoll',{safe:true}, function(err,
collection){
            //     if(err){
            //         console.log(err);
            //     }
            // });
            // 第2种连接方式
            db.createCollection('mycoll', {safe:true}, function(err,
collection){
```

```
                if(err){
                    console.log(err);
                }else{
                    //新增数据
                    // var tmp1 = {id:'1',title:'hello',number:1};
                    //          collection.insert(tmp1,
{safe:true},function(err, result){
                    //              console.log(result);
                    //          });
                    //更新数据
                    // collection.update({title:'hello'}, {$set:
{number:3}}, {safe:true}, function(err, result){
                    //     console.log(result);
                    // });
                    // 删除数据
                    // collection.remove({title:'hello'},
{safe:true},function(err,result){
                    //                   console.log(result);
                    //               });

                    // console.log(collection);
                    // 查询数据
                    var tmp1 = {title:'hello'};
                    var tmp2 = {title:'world'};
                    collection.insert([tmp1,tmp2],
{safe:true},function(err,result){
                        console.log(result);
                    });
                    collection.find().toArray(function(err,docs){
                        console.log('find');
                        console.log(docs);
                    });
                    collection.findOne(function(err,doc){
                        console.log('findOne');
                        console.log(doc);
                    });
                }

            });
            // console.log('delete ...');
            // //删除Collection
            // db.dropCollection('mycoll',
{safe:true},function(err,result){

            //           if(err){

            //         console.log('err:');
            //         console.log(err);
            //     }else{
            //         console.log('ok:');
            //         console.log(result);
```

```
            //     }
            //       });
        console.log('>>>>>>>>>>>');
        db.collection('comments',{safe:true}, function(err, collection)
{
                    if(err){
                        console.log(err);
                    }
                    collection.find({like:{$gt:
20}}).toArray(function(err,docs){
                        console.log(docs[0].like);
                    })
                });

        }else{
            console.log(err);
        }
    });
});
```