

00 启动数据库 mongodb 文档型数据库

```
1 在你安装的mongodb的位置 打开命令行
2 mongod --dbpath D:data //没有这个data文件自己建立一个
3 然后再到这个目录继续开一个cmd 然后输入mongo 连接成功后 查看 show dbs
4
5 此时打开robot3t是否看看是不是这个三个数据库
6
7
8 小技巧 拓展 可以不使用 程序员不怕稍微麻烦一丢丢
9 5. 为了方便
10 -> 新建文本文档
11 -> 写入 mongod --dbpath e:data
12 -> 把后缀改成 .bat(扩展 可不用)
13 注意: 每次你要使用 mongodb 数据库的时候, 必须要使用 mongod 执行启动服务
```

01 mongodb数据库 了解

了解 mongodb

+ 和 mysql 的区别

=> mysql 都是关系型数据库

-> 存储, 多表存储, 每一个表里面可以写一个信息, 和其他表关联

-> 多表联合查询

-> 多表之间可以存在联系, 可以使用 sql 语句让多张表联合在一起

=> mongodb 是非关系型数据库

-> 存储, 以集合(库 database)的形式存储

-> 集合里面都是以 json 文件的格式在存储

-> 多个表之间没有联系, 不能通过语句来产生联系(因为根本没有固定语句)

02 mongodb经常用到的命令行操作1

```
1 //创建并且选择（有则选择 无则建） use dbName
2 //db 查看当前数据库
3 //show dbs 查看所有数据库
4 //db.status 查看数据库状态
5 //db.getMongo() 查看数据库连接地址
6 // db.version() 查看数据库版本信息
7 //db.dropDatabase() 删除数据库 -- 慎用
8 //db.createCollection("runoob") //创建集合
9 删除集合
10 db.runoob.drop()
11 查看所有集合
12 show collections
13 查看当前集合名称
14 db.getCollectionNames()
15
16 插入单条 db.shuihu.insertOne({})
17 插入多条 db.shuihu.insertMany([{username:'徐宁',password:'5555',age:36},
18 {username:'张清',password:'5555',age:40}])
19
20
21
22 删 根据指定键值对条件
23 删除单条数据 db.colName.deleteOne({key:val})
24 例子
25 db.stu.deleteOne({"userName":"李四10"})
26 删除多条数据 db.colName.deleteMany({key:val})
27 删除所有数据 db.colName.deleteMany({})
28
29 改
30
31 根据指定键值对条件
32 修改数据 db.shuihu.updateOne({username:'张清'},{$set:{username:'张清2'}})
33 num 为正自增 负 自减
34 自增/自减单条数据
```

```
35 例子
36 db.shuihu.updateMany({age:19},{ $inc:{age:1}})
37 自增/自减单条数据
38 例子
39 db.shuihu.updateMany({age:19},{ $inc:{age:-1}})
```

03 插入数据实战

```
1 单条
2 db.shuihu.insert({username:'林冲2',password:123456,age:18 })
3 插入多条
4 例子 db.shuihu.insert([
5     username:'宋江',
6     pass:123456,
7     age:11,
8
9 ],[
10    username:'卢俊义',
11    pass:123456,
12    age:2,
13
14 ],[
15    username:'吴用',
16    pass:123456,
17    age:111,
18
19 ],[
20    username:'公孙胜',
21    pass:123456,
22    age:19,
23
24 ],[
25    username:'关胜',
26    pass:123456,
27    age:19,
28
29 ],[
30    username:'林冲',
31    pass:123456,
```

```
32   age:19,
33
34 },{
35   username:'秦明',
36   pass:123456,
37   age:15,
38
39 },{
40   username:'呼延灼',
41   pass:123456,
42   age:40,
43
44 },{
45   username:'花荣',
46   pass:123456,
47   age:66,
48
49 },{
50   username:'柴进',
51   pass:123456,
52   age:55,
53
54 },{
55   username:'李应',
56   pass:123456,
57   age:88,
58
59 }]])
60
61
62 删 根据指定键值对条件
63 删除单条数据db.shuihu.deleteOne({username:'林冲2'})
64
65 删除多条数据 db.shuihu.deleteMany({username:'林冲2'})
66 删除所有数据 db.shuihu.deleteMany({})
67
68 改
69
70 根据指定键值对条件
71 修改单条数据 db.shuihu.updateOne({username:'宋江'},{$set:{username:'宋江2'}})
72
```

```

73 修改多条数据 db.shuihu.updateMany(
74    {username:'林冲2'},
75    { $set: { username:'linchong' } }
76
77  )
78
79 num 为正自增 负 自减
80 自增/自减单条数据 db.shuihu.updateOne({username:'linchong'},{$inc:
{age:1}})
81  db.shuihu.updateOne({username:'linchong'},{$inc:{age:-1}})
82

```

04mongodb数据库 常用命令3 修改

```

1  1 基本查询所有数据
2  db.shuihu.find()
3
4
5  2 格式化查询所有数据 更加美观
6  db.shuihu.find().pretty()
7
8
9  3 指定键值对条件查询
10 db.shuihu.find({key:val})
11
12
13 4 指定条件查询(可以为{}表示所有数据) 并限制字段显示
14 //1 inclusion模式 指定返回的键, 不返回其他键
15 db.shuihu.find({username:'张清2'}, {username:1,password:1})
16
17
18 //2 db.exclusion 模式 指定不返回的键 返回其他键
19 db.shuihu.find({username:'张清2'}, {username:0,password:0})
20
21
22 5 分页查询 分页可以直接交给后端来做
23 db.colName.find({key:val}).limit(num).skip(start)
24 //num 表示个数

```

```

25 //start 表示开始索引 默认0
26 例子 db.shuihu.find({}).limit(5).skip(2)
27
28
29 6 排序查询
30 db.shuihu.find({}).sort({age:1})
31
32 // 1 升序 -1 降序
33 // 例子
34 db.shuihu.find({}).sort({age:-1})
35
36
37
38
39 7 区间查询
40 // 小于val1 大于val2
41
42 例子 db.shuihu.find({age:{$lt:88,$gt:33}})
43 // 小于等于val1 大于等于val2
44
45 例子 db.shuihu.find({age:{$lte:88,$gte:33}})

```

05 mongodb 常用命令3 数据查询

```

1 1 模糊查询 -- 实战中用于搜索
2 db.shuihu.find({username:/花/}) // 查看key中包含val的数据
3
4
5
6 db.shuihu.find({username:/^花/}) //查询key中包含val 以val开头的的数据
7
8
9
10
11
12

```

```
13
14 2 或查询
15 db.shuihu.find({$or:[{username:'花荣'},{username:'林冲'}]})
16
17 3 且查询
18 db.shuihu.find({pass:123456,age:20} )
19
20
21
22
```

5.5 mongodb中的顶级算法 objectid

```
1 //查询方式 db.shuihu.find({_id:(ObjectId("5fa15a5faaf2eb745bb303de"))})
```

06 node操作mongodb

```
1 我们先进入01的文件夹 然后npm init 一路回车 创建一个package.json 文件
2 方便记录我们的依赖
3
4 然后我们再安装mongoose 这是一个数据库的框架 帮我们处理了 数据库很多细节问题
5 mongoose优点！
6 可以为文档创建一个模式结构（Schema）
7 可以对模型中的对象进行验证
8 数据可以通过类型转换为对象模型
9 可以使用中间件来与业务逻辑挂钩
10 比Node原生的mongodb驱动更容易
11
12
13
14 此时我们再文件夹新建一个db.js db.js的意思是database 就是数据库的意思 所以叫做db
15
16 引入mongoose 模块
17 //引入mongoose 模块
18 const mongoose = require('mongoose')
19 //2 连接mongodb并选择指定数据库 dbName
```

```

20 mongoose.connect('mongodb://localhost:27017/modu')
21 //3 连接成功
22 mongoose.connection.on('connected', ()=>{
23     console.log('我在监听成功状态')
24 })
25 //4 连接断开
26 mongoose.connection.on('disconnected', ()=>{
27     console.log('我在监听连接断开状态')
28 })
29 //5 连接错误
30 mongoose.connection.on('error', ()=>{
31     console.log('我在监听连接错误状态')
32 })
33 //6 连接成功之后 将模块暴露出来
34 module.exports = {
35     mongoose:mongoose
36 }
37
38
39
40 现在我们需要先打开cmd mongo bin目录下第一次连接池 mongod --dbpath D:data
41 第二部分 不用mongo了 用我们的代码连接 node db.js 看看效果
42
43

```

此时已经一切成功准备就绪了

我们看此时命令行的状态

```

D:\千峰备课\最新\noddemongodday3\v1>node db.js
(node:39616) DeprecationWarning: current URL string parser is deprecated, and will be removed in a future version. To use the new parser, pass option { useNewUrlParser: true } to MongoClient.connect.
(node:39616) DeprecationWarning: current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor.

```


其实数据库此时是使用测试时完全没有问题的 这两个东西 是旧版本有新特性更改 还有一个可能未来某个时刻被移除 让你加上这两个属性 这时候 我们 把mongoose代码变成这个样子 这个小警告就结束了

```
1 //引入mongoose 模块
2 const mongoose = require('mongoose')
3 //2 连接mongodb并选择指定数据库 dbName
4 mongoose.connect('mongodb://localhost:27017/modu',
5 { useNewUrlParser: true,useUnifiedTopology: true } )
6 //3 连接成功
7 mongoose.connection.on('connected', ()=>{
8     console.log('我在监听成功状态')
9 })
10 //4 连接断开
11 mongoose.connection.on('disconnected', ()=>{
12     console.log('我在监听连接断开状态')
13 })
14 //5 连接错误
15 mongoose.connection.on('error', ()=>{
16     console.log('我在监听连接错误状态')
17 })
18 //6 连接成功之后 将模块暴露出来
19 module.exports = {
20     mongoose:mongoose
21 }
22
23
```

此时的小警告已经消失了 这个坑如果工作中遇到 先查官方文档 再查百度 谷歌 实在不行自己试试 看加在哪里 去踩这个坑 程序中这种配置的坑很多 只能是这种方式去踩 这个不是逻辑代码