00 启动数据库 mongodb 文档型数据库

```
在你安装的mongodb的位置 打开命令行
mongod --dbpath D:data //没有这个data文件自己建立一个
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

此时打开robot3t是否看看是不是这个三个数据库


小技巧 拓展 可以不使用 程序员不怕稍微麻烦一丢丢
5. 为了更方便
  -> 新建文本文档
  -> 写入 mongod --dbpath e:data
  -> 把后缀改成 .bat(扩展 可不用)
  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

```
//创建并且选择 (有则选择 无则建) use dbName
//db 查看当前数据库
//show dbs 查看所有数据库
//db.status 查看数据库状态
//db.getMongo() 查看数据库连接地址
// db.version() 查看数据库版本信息
//db.dropDatabase() 删除数据库 --慎用
//db.createCollection("runoob") //创建集合
 删除集合
  db.runoob.drop()
  查看所有集合
  show collections
  查看当前集合名称
  db.getCollectionNames()

  插入单条 db.shuihu.insertOne({})
  插入多条 db.shuihu.insertMany([{username:'徐宁',password:'5555',age:36},{username:'张清',password:'5555',age:40}])




  删 根据指定键值对条件
删除单条数据 db.colName.deleOne({key:val})
例子
db.stu.deleteOne({"userName":"李四10"})
删除多条数据 db.colName.deleteMany({key:val})
删除所有数据 db.colName.deleteMany({})

改

根据指定键值对条件
修改数据 db.shuihu.updateOne({username:'张清'},{$set:{username:'张清2'}})
num 为正自增 负 自减
自增/自减单条数据
```

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

03 插入数据实战

```
单条
db.shuihu.insert({username:'林冲2',password:123456,age:18 })
插入多条
例子 db.shuihu.insert([{
 username:'宋江',
 pass:123456,
 age:11,

},{
  username:'卢俊义',
  pass:123456,
  age:2,

 },{
  username:'吴用',
  pass:123456,
  age:111,

 },{
  username:'公孙胜',
  pass:123456,
  age:19,

 },{
  username:'关胜',
  pass:123456,
  age:19,

 },{
  username:'林冲',
  pass:123456,
```

```
 age:19,

},{
 username:'秦明',
 pass:123456,
 age:15,

},{
 username:'呼延灼',
 pass:123456,
 age:40,

},{
 username:'花荣',
 pass:123456,
 age:66,

},{
 username:'柴进',
 pass:123456,
 age:55,

},{
 username:'李应',
 pass:123456,
 age:88,

}])


删 根据指定键值对条件
删除单条数据db.shuihu.deleteOne({username:'林冲2'})

删除多条数据 db.shuihu.deleteMany({username:'林冲2'})
删除所有数据 db.shuihu.deleteMany({})

改

根据指定键值对条件
修改单条数据 db.shuihu.updateOne({username:'宋江'},{$set:{username:'宋江2'}})

```

```
修改多条数据 db.shuihu.updateMany(
    {username:'林冲2'},
    { $set: { username:'linchong' }  }

 )

num 为正自增 负 自减
自增/自减单条数据 db.shuihu.updateOne({username:'linchong'},{$inc:{age:1}})
   db.shuihu.updateOne({username:'linchong'},{$inc:{age:-1}})

```

04mongodb数据库 常用命令3 修改

```
1 基本查询所有数据
db.shuihu.find()


2 格式化查询所有数据 更加美观
db.shuihu.find().pretty()


3 指定键值对条件查询
db.shuihu.find({key:val})


4 指定条件查询(可以为{}表示所有数据) 并限制字段显示
//1 inclusion模式 指定返回的键，不返回其他键
db.shuihu.find({username:'张清2'},{username:1,password:1})


//2 db.exclusion 模式 指定不返回的键 返回其他键
db.shuihu.find({username:'张清2'},{username:0,password:0})


5 分页查询 分页可以直接交给后端来做
db.colName.find({key:val}).limit(num).skip(start)
//num 表示个数
```

```
//start 表示开始索引 默认0
例子 db.shuihu.find({}).limit(5).skip(2)


6 排序查询
db.shuihu.find({}).sort({age:1})

// 1 升序 -1 降序
// 例子
db.shuihu.find({}).sort({age:-1})



7 区间查询
// 小于val1 大于val2

例子 db.shuihu.find({age:{$lt:88,$gt:33}})
// 小于等于val1 大于等于val2

例子 db.shuihu.find({age:{$lte:88,$gte:33}})
```

05 mongodb 常用命令3 数据查询

```
1 模糊查询 -- 实战中用于搜索
db.shuihu.find({username:/花/}) // 查看key中包含val的数据


db.shuihu.find({username:/^花/}) //查询key中包含val 以val开头的数据



```

```
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"))})
```

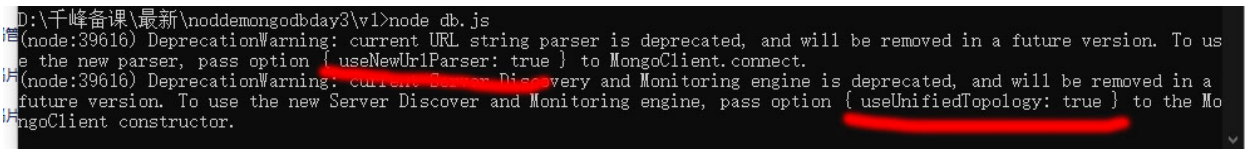
06 node操作mongodb

```
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
```

```
mongoose.connect('mongodb://localhost:27017/modu')
//3 连接成功
mongoose.connection.on('connected', ()=>{
    console.log('我在监听成功状态')
})
//4 连接断开
mongoose.connection.on('disconnected',()=>{
    console.log('我在监听连接断开状态')
})
//5 连接错误
mongoose.connection.on('error',()=>{
    console.log('我在监听连接错误状态')
})
//6 连接成功之后 将模块暴露出来
module.exports = {
    mongoose:mongoose
}



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
第二部分 不用mongo了 用我们的代码连接 node db.js 看看效果


```

此时已经一切成功准备就绪了

我们看此时命令行的状态



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

```
//引入mongoose 模块
const mongoose = require('mongoose')
//2 连接mongodb并选择指定数据库 dbName
mongoose.connect('mongodb://localhost:27017/modu',
{ useNewUrlParser: true,useUnifiedTopology: true } )
//3 连接成功
mongoose.connection.on('connected', ()=>{
    console.log('我在监听成功状态')
})
//4 连接断开
 mongoose.connection.on('disconnected',()=>{
     console.log('我在监听连接断开状态')
 })
 //5 连接错误
 mongoose.connection.on('error',()=>{
     console.log('我在监听连接错误状态')
 })
 //6 连接成功之后 将模块暴露出来
 module.exports = {
     mongoose:mongoose
 }


```

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