<https://www.runoob.com/mongodb>

# 关系型数据库和非关系型数据库

## 关系型数据库 mysql sqlsever

表就是关系 ，或者表与表之间存在关系

* 所有关系型数据库都是通过sql语句操作的
* 所有关系型数据库的操作之前都需要设计表结构
* 而且数据表支持约束 唯一，主键，默认值，非空，外键

## 非关系型数据库 mongoDB

* 非关系型数据库是非常灵活的
* 有的非关系型数据库就是key-value对
* 但是MongoDB是长得最象关系型数据库的非关系型数据库

数据库 =》 数据库

数据表 = 》集合（数组）

表记录 = 》（文档对象）

* MongoDB不需要设计表结构
* 也就是说你可以任意的往里面存数据，没有表结构性这么一说
* 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。

# 安装

<https://www.mongodb.com/download-center/community>

配置环境变量 ：C:\Program Files\MongoDB\Server\3.4\bin

mongod --version 查看安装是否成功

# 开启和关闭数据库

## 启动 mongod

在d盘下面创建一个data文件，data文件中创建db文件，即为存储数据的目录

如果想修改默认目录：

Mongod --dbpath 数据存储路径

不要关闭cmd

## 关闭

Ctrl+C关闭

# 连接和退出数据库

## 连接：Mongo 该命令默认连接本机的mongoDB

## 退出 ：exit

# 基本命令

## show dbs 查看所有数据库

## db 查看当前操作的数据库

## use 数据库名称 切换到指定数据（如果没有会自动创建）

## db.表名.insertOne() 添加数据

## show collections 查看集合(表)

## db.表名.find() 查询所有表中的数据

# 在Node中操作MongoDB数据库

## 使用官方mongodb包操作

<https://www.npmjs.com/package/mongodb>

## 使用第三方mongoose来操作MongoDB数据库

第三方包：mongoose基于MonggoDB官方是mongodb包再一次做了封装

<http://www.mongoosejs.net/>

## MongoDB数据库基本概念

数据库

集合（数据表）

文档（数据）

* 一个数据库可以有多个集合（表）
* 一个表中可以有多个文档（数据）
* 不需要和mysql一样建库，建表，设计表结构，
* 当你要插入数据值，直接指定往哪个数据库哪个集合插入就行

{

    qq数据库：{

        user集合；[

            //数据以键值对的形式

            {name:'xxx',age:19},

            {name:'xxx',age:19}

        ],

        products集合：[

            {},

            {}

        ]

    }，

    baidu数据库{

    }

}

## Mongoose

### 安装

npm install mongoose

### 起步

//导包

var mongoose = require("mongoose");

//连接mongoDB数据库 info为数据库

mongoose.connect('mongodb://localhost/info',{ useMongoClient:true })

//创建一个模型

//就是设计数据库

//mongoDB是动态的，只需要在代码中设计数据库就可以

// mongoose这个包就可以让你的设计编写过程变得非常简单

// var cat = mongoose.model('表名',{数据名：数据类型})

var Cat = mongoose.model('Cat',{name:String});

for(var i=0;i<20;i++){

    //实例化一个student

    var shili = new  Cat({name:'xiaoxiao'+i});

    //持久化保存sgili实例

    shili.save(function(err){

        if(err){

            // console.log(err)

        }else{

            console.log("sussess")

        }

    })

}

### 步骤

* 导mongoose包

var mongoose = require(‘mongoose’)

* 连接数据库 指定的数据库不需要存在，在第一条插入数据之后会自动被创建出来mongoose.connect(‘mongodb://localhost/数据库名称’)
* 设计集合结构（表结构） 约束的目的是为了使数据更加完整

var userSchema = new schema({name:String,age:{type:number,required:true} })

* 将文档架构发布为模型 mongoose.model用来将一个架构发布为model（模型）

第一个参数：传入一个大写的名词单数字符，用来表示你的数据表名称

mongoose会自动将大写名称的字符串生成小写复数的集合（表）名称

第二个参数：架构的Schema

返回值：模型构造函数

var User = mongoose.model(‘集合名称（表名）’，userSchema)

### 增删改查操作

##### 增加数据

var admin = new User({

    user:'admin',

    pasd:'123456',

    email:'admin@qq.com'

})

admin.save(function(err,ret){

    if(err){

        console.log(err)

    }else{

        console.log('插入成功')

        console.log(ret)//插入的数据

    }

})

##### 查询数据

###### //查询所有数据

User.find(function(err,ret){

    if(err){

        console.log('查询失败')

    }else{

        console.log('查询成功')

        console.log(ret)//插入的数据

    }

})

###### //按条件查

//查询前1条数据 limit(查询的数据量)

User.find(function(err,ret){

    if(err){

        console.log('查询失败')

    }else{

        console.log('查询成功')

        console.log(ret)//插入的数据

    }

}).limit(1)

User.findOne({

    user:'admin'

},function(err,ret){

    if(err){

        console.log('查询失败')

    }else{

        console.log('查询成功')

        console.log(ret)//插入的数据

    }

})

//查询user = zs，且pasd='123456

User.find({

    pasd:'123456'

},function(err,ret){

    if(err){

        console.log('查询失败')

    }else{

        console.log('查询成功')

        console.log(ret)//插入的数据

    }

}).where('user').equals('zs')

//1.以JSON对象的形式

User.find({

occupation: /host/,

'name': 'Ghost',

age: { $gt: 17, $lt: 66 },

likes: { $**in**: ['vaporizing', 'talking'] }

}).

limit(10).

sort({ occupation: -1 }).

select({ name: 1, occupation: 1 }).

exec(callback);

// 2.使用query 形式

User. find({ occupation: /host/ }).

where('name).equals('Ghost'). //name=Ghost

where('age').gt(17).lt(66). // 17<age<66

where('likes').in(['vaporizing', 'talking']). //likes ='vaporizing'ikes ='talking'

limit(10). //前10条信息

sort('-occupation'). //按occupation排序

select('name occupation'). //查询name 和 occupation字段

exec(callback); //结束

###### 根据id查询 findById()

User.findById(id,[callback])

##### 删除数据

###### 删除符合的一条deleteOne

User.deleteOne({

    pasd:'123456'

},function(err,ret){

    if(err){

        console.log('删除失败')

    }else{

        console.log('删除成功')

        console.log(ret)//插入的数据

    }

})

###### 删除符合条件的所有数据 deleteMany()

###### 根据ID删除 findIdAndDelete()

User.findByIdAndDelete(id,[options],[callback])

##### 修改数据

###### 根据id修改findIdAndUpdate( id,数据,[callback])

User.findByIdAndUpdate('5e743d2ff071a72058387032',{

    pasd:'zdc'

},function(err,ret){

    if(err){

        console.log('更新失败')

    }else{

        console.log('更新成功')

        // console.log(ret)//插入的数据

    }

})