笔记 5

MongoDB数据库基本操作

启动和关闭数据库

- 启动：

```
# mongodb 默认使用执行 mongod 命令所处盘符根目录下的 /data/db 作为自己的数据存储目录
# 所以在第一次执行改命令之前需要自己手动创建一个 /data/db 目录
mongod
```

- 如果想要修改默认的数据存储路径，可以

```
mongod --dbpath=数据存储目录路径
```

- 停止

```
在服务器上 Ctrl+c 或者直接关闭 cmd 窗口
```

连接和退出数据库

- 连接

```
# 该命令默认连接本机的 mongoDB 服务
mongo
```

- 退出

```
exit
```

基本命令

- `show dbs`
  - 查看所以数据库列表
- `db`
  - 查看当前操作的数据库
- `use 数据库名称`
  - 切换到指定数据库(如果没有会新建)

- 具体命令点击查看

使用 Node 操作 MongoDB 数据

- 可以使用官方的包
  - 官方网址
- 也可以使用第三方 mongoose 来操作
  - 网址

mongoose 使用

1.起步

```
npm i mongoose
```

hello world:

```
var mongoose = require('mongoose');

// 连接数据库
mongoose.connect('mongodb://localhost/test', {
    useNewUrlParser: true
});

mongoose.Promise = global.Promise;
// 创建一个模型
// 就是再设计数据库
// MongoDB 是动态的，非常灵活，只需要在代码中设计你的数据库就可以了
// mongoose 这个包就可以让你的设计编写过程变得非常的简单
// 创建一个 集合：名为 cat，里面有个对象叫 name ，数据类型是 string
var Cat = mongoose.model('Cat', {
    name: String
});

// 实例化一个 cat
var kitty = new Cat({
    name: '喵喵喵'
});

// 持久化保存 Kitty 实例
kitty.save(function (err) {
    if (err) {
        console.log(err);
    } else {
        console.log('hello');
    }
})
```

MongoDB 数据库基本概念

- 可以有多个 数据库
- 一个数据库中可以有多个 集合 (表)
- 一个集合中可以有多个 文档 (表记录)
- 文档很灵活，没有任何限制
- MongoDB 很灵活，不需要像 MySQL 一样先创建数据库、表、设计表结构
    - 这里只需要：当你需要插入数据的时候，只需要指定哪个数据库的哪个集合操作就可以了
    - 一切都由 MongoDB 来帮你自动完成建库建表这一系列操作

MongoDB 的增删改查

具体配置

官方指南

```
// 引入 mongoose 包
var mongoose = require('mongoose')
// 拿到 Schema 架构对象
var Schema = mongoose.Schema;
// 1. 连接数据库
// 连接的数据库不需要存在，再插入第一条数据之后就会自动创建数据库
mongoose.connect('mongodb://localhost/pets', {
    useNewUrlParser: true
})

// 2. 设计集合结构(表结构)
var petsSchema = new Schema({
    petName: {
        type: String,
        required: true
    },
    price: {
        type: Float32Array,
        required: true
    },
    explain: {
        type: String
    }
})

// 3. 将文档架构发布为模型
// mongoose.model 方法就是用来将一个架构发布为 model
// 第一个参数：传入一个大写名词单数字符串用来表示你的数据库名称
//       mongoose 会自动将大写名称的字符串生成 小写复数 的集合名称
//       例如这里 'Pet' 会被转为 pets
// 第二个参数：架构 Schema
// 返回值 ：  模型构造函数
var Pet = mongoose.model('Pet', petsSchema)
```

增加：

```
var pets = new Pet({
    petName:'旺财',
    price:'500',
    explain:'这狗很乖的！'
})

pets.save(function(err,ret) {
    if(err) {
        console.log('保存失败！');
    }else {
        console.log('保存成功！');
        console.log(ret);
    }
})
```

查询

查询所有

```
Pet.find(function (err,DBdata) {
    if (err) {
        console.log('查询失败！');
    } else {
        console.log('查询成功！');
        console.log(DBdata);
    }
})
```

条件查询

```
Pet.find({
    // 放条件 可放多个
    petName : '旺财'
    // ,price : '500'
},function (err,DBdata) {
    if (err) {
        console.log('条件查询失败！');
    } else {
        console.log('条件查询成功！');
        console.log(DBdata);
    }
})
```

条件查询单个数据对象

```
Pet.findOne({
    petName : '哈士奇'
},function (err,DBdata) {
    if (err) {
        console.log('条件查询单个数据对象失败! ');
    } else {
        console.log('条件查询单个数据对象成功! ');
        console.log(DBdata);
    }
})
```

删除

```
Pet.remove({
    _id : '5cb817a798c2be176423d1ac'
},function (err,DBdata) {
    if (err) {
        console.log('删除失败! ');
    } else {
        console.log('删除成功! ');
    }
})
```

更新

```
Pet.findByIdAndUpdate('5cb817d53d6aa115fc52ab18',{
    explain : '这就是土狗啊! '
},function (err,DBdata) {
    if (err) {
        console.log('更新失败! ');
    } else {
        console.log('更新成功! ');
    }
})
```

更多看官方文档 API