持久化mongodb

资源

- mongodb相关
  - MonoDB：下载
  - node驱动：文档
  - mongoose:文档
- redis相关
  - redis：下载
  - node_redis:文档
- 可视化工具：Robo3T

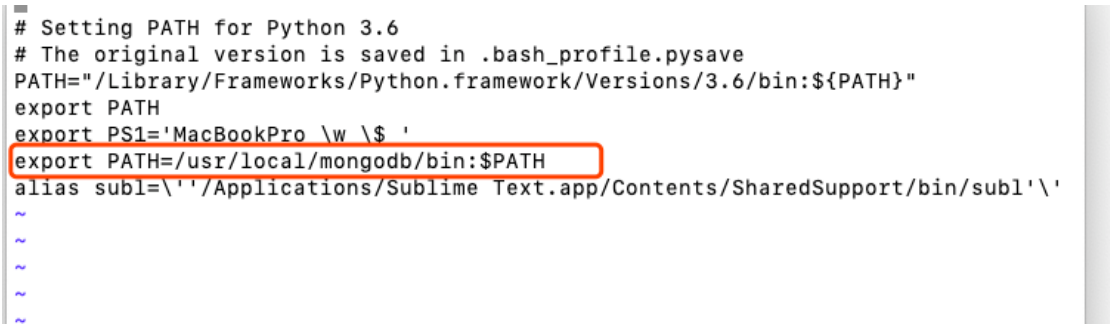
mongodb安装、配置

- 下载安装
- 创建data/db文件夹
    - 系统根目录下创建文件夹

    ```
    sudo mkdir -p /data/db
    ```

    - 给 /data/db 文件夹赋予权限

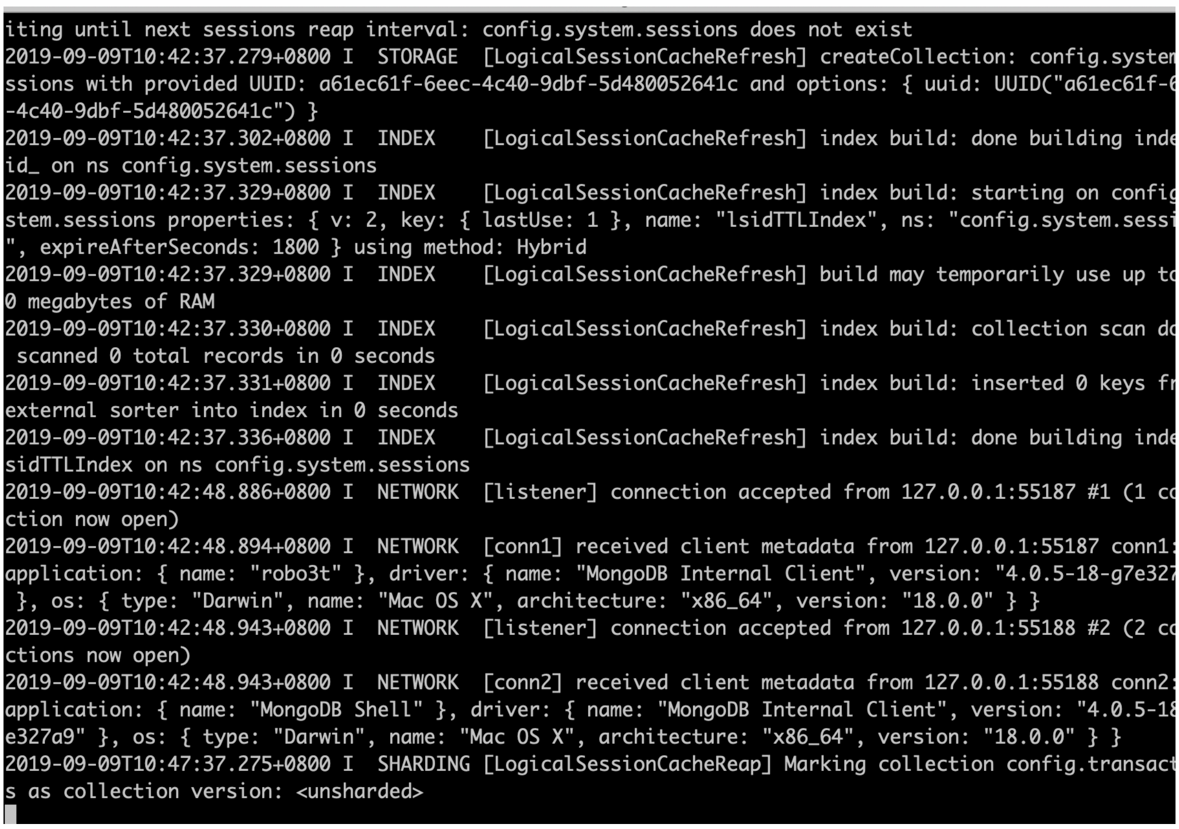
    ```
    sudo chown xxx(用户名) /data/db
    ```

- 配置环境变量
    - vim ~/.bash_profile
    - 

    ```
    # Setting PATH for Python 3.6
    # The original version is saved in .bash_profile.pysave
    PATH="/Library/Frameworks/Python.framework/Versions/3.6/bin:${PATH}"
    export PATH
    export PS1='MacBookPro \w \$ '
    export PATH=/usr/local/mongodb/bin:$PATH
    alias subl=\''/Applications/Sublime Text.app/Contents/SharedSupport/bin/subl'\'
    ~
    ~
    ~
    ~
    ~
    ```

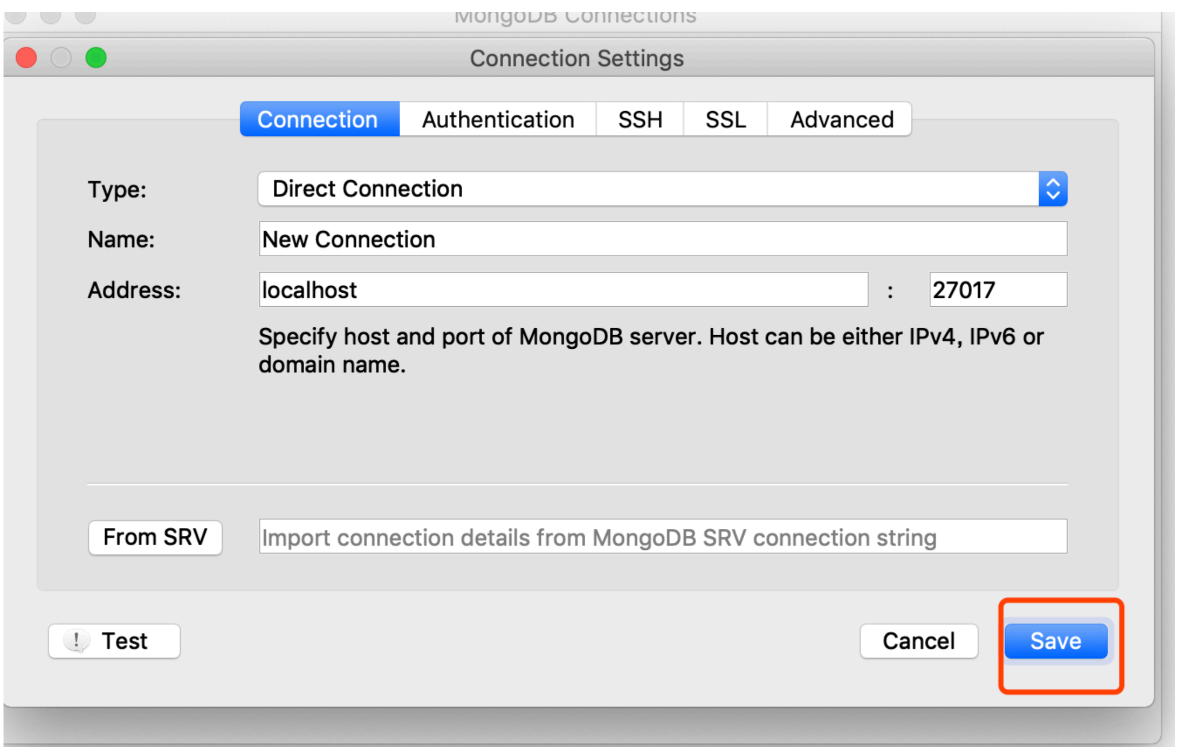
- 启动

```
mongod
```



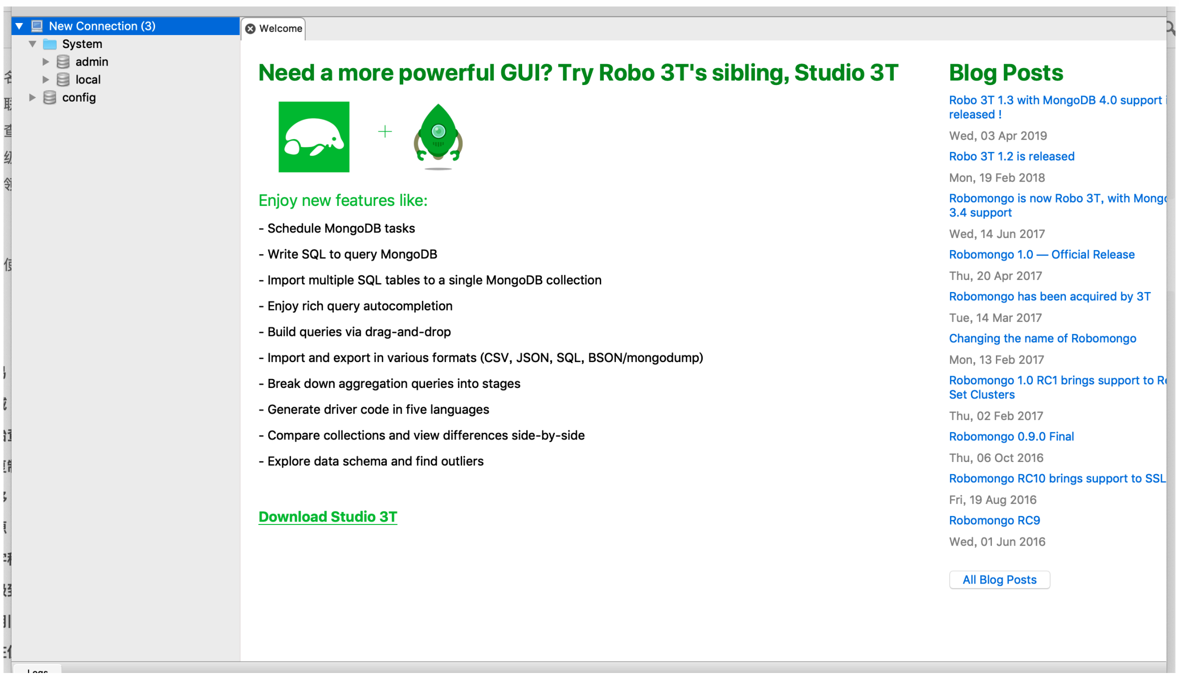
```
iting until next sessions reap interval: config.system.sessions does not exist
2019-09-09T10:42:37.279+0800 I  STORAGE  [LogicalSessionCacheRefresh] createCollection: config.systen
ssions with provided UUID: a61ec61f-6eec-4c40-9dbf-5d480052641c and options: { uuid: UUID("a61ec61f-6
-4c40-9dbf-5d480052641c") }
2019-09-09T10:42:37.302+0800 I  INDEX    [LogicalSessionCacheRefresh] index build: done building inde
id_ on ns config.system.sessions
2019-09-09T10:42:37.329+0800 I  INDEX    [LogicalSessionCacheRefresh] index build: starting on config
stem.sessions properties: { v: 2, key: { lastUse: 1 }, name: "lsidTTLIndex", ns: "config.system.sessi
", expireAfterSeconds: 1800 } using method: Hybrid
2019-09-09T10:42:37.329+0800 I  INDEX    [LogicalSessionCacheRefresh] build may temporarily use up to
0 megabytes of RAM
2019-09-09T10:42:37.330+0800 I  INDEX    [LogicalSessionCacheRefresh] index build: collection scan do
 scanned 0 total records in 0 seconds
2019-09-09T10:42:37.331+0800 I  INDEX    [LogicalSessionCacheRefresh] index build: inserted 0 keys fr
external sorter into index in 0 seconds
2019-09-09T10:42:37.336+0800 I  INDEX    [LogicalSessionCacheRefresh] index build: done building inde
sidTTLIndex on ns config.system.sessions
2019-09-09T10:42:48.886+0800 I  NETWORK  [listener] connection accepted from 127.0.0.1:55187 #1 (1 co
ction now open)
2019-09-09T10:42:48.894+0800 I  NETWORK  [conn1] received client metadata from 127.0.0.1:55187 conn1:
application: { name: "robo3t" }, driver: { name: "MongoDB Internal Client", version: "4.0.5-18-g7e327
 }, os: { type: "Darwin", name: "Mac OS X", architecture: "x86_64", version: "18.0.0" } }
2019-09-09T10:42:48.943+0800 I  NETWORK  [listener] connection accepted from 127.0.0.1:55188 #2 (2 co
ctions now open)
2019-09-09T10:42:48.943+0800 I  NETWORK  [conn2] received client metadata from 127.0.0.1:55188 conn2:
application: { name: "MongoDB Shell" }, driver: { name: "MongoDB Internal Client", version: "4.0.5-18
e327a9" }, os: { type: "Darwin", name: "Mac OS X", architecture: "x86_64", version: "18.0.0" } }
2019-09-09T10:47:37.275+0800 I  SHARDING [LogicalSessionCacheReap] Marking collection config.transact
s as collection version: <unsharded>
```

- 连接
- 打开Robo 3T可视化工具



Type: Direct Connection
Name: New Connection
Address: localhost : 27017
Specify host and port of MongoDB server. Host can be either IPv4, IPv6 or domain name.
Import connection details from MongoDB SRV connection string

-
-



Need a more powerful GUI? Try Robo 3T's sibling, Studio 3T

Enjoy new features like:

- Schedule MongoDB tasks
- Write SQL to query MongoDB
- Import multiple SQL tables to a single MongoDB collection
- Enjoy rich query autocompletion
- Build queries via drag-and-drop
- Import and export in various formats (CSV, JSON, SQL, BSON/mongodump)
- Break down aggregation queries into stages
- Generate driver code in five languages
- Compare collections and view differences side-by-side
- Explore data schema and find outliers

Download Studio 3T

Blog Posts

Robo 3T 1.3 with MongoDB 4.0 support released !
Wed, 03 Apr 2019
Robo 3T 1.2 is released
Mon, 19 Feb 2018
Robomongo is now Robo 3T, with Mongo 3.4 support
Wed, 14 Jun 2017
Robomongo 1.0 — Official Release
Thu, 20 Apr 2017
Robomongo has been acquired by 3T
Tue, 14 Mar 2017
Changing the name of Robomongo
Mon, 13 Feb 2017
Robomongo 1.0 RC1 brings support to Re Set Clusters
Thu, 02 Feb 2017
Robomongo 0.9.0 Final
Thu, 06 Oct 2016
Robomongo RC10 brings support to SSL
Fri, 19 Aug 2016
Robomongo RC9
Wed, 01 Jun 2016

All Blog Posts

MongodDB参考文档

monogodb原生驱动

- 安装mongodb模块：`npm i mongodb --save`
- 连接mongodb

```
//客户端
const MongoClient = require('mongodb').MongoClient;

// 链接的url
const url = 'mongodb://localhost:27017';

// 数据库的名字
const dbName = 'school';

(async function () {
    // 0.创建客户端
    const client = new MongoClient(url, { useNewUrlParser: true });
    // 1.链接数据库
    await client.connect();
    console.log("Connected successfully to server");

    // 2.获取数据库
    const db = client.db(dbName);

    //3. 获取集合
    const grade1 = db.collection('grade1');
    await grade1.deleteMany();
    //4. 插入文档 insertOne()插入一条数据 insetMany()插入多条数据
    let r = grade1.insertMany([
        { name: "张三", age: 20, hobby: ['吃饭', '睡觉', '打豆豆'] },
        { name: "李四", age: 40, hobby: ['妹子', '篮球'] },
        { name: "王五", age: 20, hobby: ['妹子', '睡觉'] },
        { name: "赵六", age: 16, hobby: ['妹子'] },
        { name: "小马哥", age: 18, hobby: ['篮球'] }
    ]);
    console.log('插入成功', r)
    //5. 查询文档 findOne()查询一条数据 find({})查询所有数据  也可以定义条件
    //查询name='大王'
    r = await grade1.find({ name: '张三' }).toArray();
    console.log('查询结果', r)
    //6. 更新文档
    r = await grade1.updateOne({ name: '李四' }, { $set: { name: '王五' } });
    console.log('更新成功', r.result);
    // const doc4 = await grade1.deleteOne({ name: "张三" });
    // console.log('删除成功', doc4.result);
    client.close();
})()
```

操作符

常用的有以下内容,具体的请查看操作符文档

```
r = await grade1.find({ name: '张三' }).toArray();
// 比较运算符  $gt大于  $lt 小于 $gte:大于等于 $lte:小于等于 $in:[10,20] 在10到20之间 $nin:[10,20] 不在10到20之间的
r = await grade1.find({age:{$in:[20,40]}}).toArray();

// 逻辑运算 $and $not $nor $or $ne 不等于

// 查询姓名叫王五并且年龄为20岁的数据
```

```
r = await grade1.find({name:'王五',age:20}).toArray();
// 查询姓名叫张三或者年龄为20岁的数据
r = await grade1.find({ $or: [{ name: '张三'},{age: 20 }] }).toArray();

// 查询年龄不大于20且age不小于16的人员
r = await grade1.find({
    $nor:[
        {
            age:{
                $gt:20
            }
        },
        {
            age:{
                $lt:16
            }
        }
    ]
}).toArray();

// 正则表达式
r = await grade1.find({name:{$regex:/^张/}}).toArray();

// $all 查找指定字段包含所有指定内容的数据
r = await grade1.find({ hobby: { $all: ['妹子'] } }).toArray();
// $in  查找指定字段只有指定内容其一的数据
```

```
r = await grade1.find({ hobby: { $in: ['妹子','睡觉'] } }).toArray();
// $size 查找指定字段的数据有三条的
r = await grade1.find({ hobby: { $size:3 } }).toArray();


// cursor的方法
// 查询前两条
r = await grade1.find().limit(2).toArray();
// 跳过前2条数据
r = await grade1.find().skip(2).toArray();
// 根据age字段进行排序,1表示正序 -1表示倒序
r = await grade1.find().sort({age:-1}).toArray();

//分页
const pageIndex = 1;
const pageSize = 3;
r = await grade1.find().skip((pageIndex - 1) * pageSize).limit(pageSize).sort({name:1}).toArray();

// forEach方法 和map方法
r.forEach( ele => {
    console.log(ele);
});
//查询所有学生的姓名
let names = r.map(ele=>ele.name);
console.log(names);//[ '张三', '李四', '王五', '赵六', '小马哥' ]
```

```

//创建文本索引 有助于加速查询

//索引通常能够极大的提高查询的效率，如果没有索引，MongoDB在读取数据时必须扫描集合中的每个文件并选取那些符合查询条件的记录。

//这种扫描全集合的查询效率是非常低的，特别在处理大量的数据时，查询可以要花费几十秒甚至几分钟，这对网站的性能是非常致命的。

//索引是特殊的数据结构，索引存储在一个易于遍历读取的数据集合中，索引是对数据库表中一列或多列的值进行排序的一种结构

//backrgound:可以指定以后台方式创建索引
///{unique:true} 创建唯一索引,查询速度更快
await grade1.createIndex({ name: "text" }, {background:true}); //创建text索引 一个集合最多只能有一个text索引
r = await grade1.find({$text:{$search:'张三'}}).toArray(); //按词


grade1.insertOne({ hisScore: [60, 80, 100] });
// 历史成绩有没有出现在70到90之间
r = await grade1.find({ hisScore: { $elemMatch: { $gt: 70, $lt: 90 } } }).toArray();


```

```
//MongoDB中聚合(aggregate)主要用于处理数据(诸如统计平均值,求和等), 并返回计算后的数据结果。有点类似sql语句中的 count(*)
// 聚合  $sum 计算和 $avg 计算平均值  $min:获取集合中所有文档对应值的最小值
// $first 根据资源文档的排序获取第一个文档数据
// $last 根据资源文档的排序获取最后一个文档数据
// select age,count(1) from grade1 group by age;
r = await grade1.aggregate([{ $group: { _id: "$age", count: { $sum: 1 } } }]).toArray();
r = await grade1.aggregate([{ $group: { _id: "$age", avgScore: { $avg: '$score' } } }]).toArray();
r = await grade1.aggregate([{ $group: { _id: "$age", minScore: { $min: '$score' }, count: { $sum: 1 } } }]).toArray();
r = await grade1.aggregate([{ $group: { _id: "$age", minScore: { $first: '$name' }, count: { $sum: 1 } } }]).toArray();
r = await grade1.aggregate([{ $group: { _id: "$age", minScore: { $last: '$name' }, count: { $sum: 1 } } }]).toArray();

//6. 更新文档
r = await grade1.updateOne({ name: '李四' }, { $set: { name: '王五' } });
console.log('更新成功', r.result);
// const doc4 = await grade1.deleteOne({ name: "张三" });
// console.log('删除成功', doc4.result);
```

地理位置获取

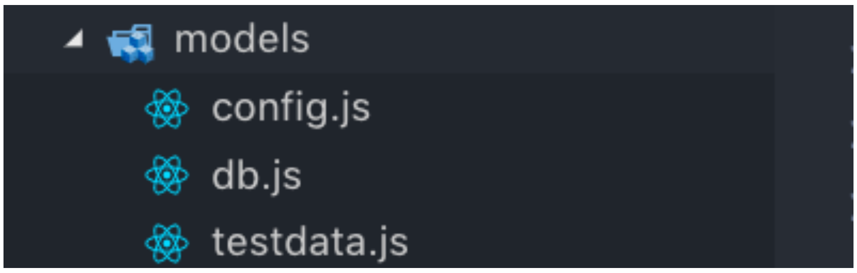
案例:搜索天安门附近地铁站的地铁

```
//客户端
const MongoClient = require('mongodb').MongoClient;

// 链接的url
const url = 'mongodb://localhost:27017';

// 数据库的名字
const dbName = 'classes';

(async function () {
    // 0.创建客户端
    const client = new MongoClient(url, { useNewUrlParser: true });
    // 1.链接数据库
    await client.connect();
    console.log("Connected successfully to server");

    // 2.获取数据库
    const db = client.db(dbName);

    //3. 获取集合
    const stations = db.collection('stations');
    await stations.deleteMany();
    await stations.insertMany([
        { name: '天安门东', loc: [116.397128, 39.916527] },
        { name: '天安门西', loc: [116.39805, 39.913776] },
        { name: '王府井', loc: [116.418668, 39.922232] },
    ]);
    await stations.createIndex({ loc: '2dsphere' })
    r = await stations.find({
        loc: {
            $nearSphere: {
                $geometry: {
                    type: "Point",
                    coordinates: [116.403847, 39.915526]
                },
                $maxDistance: 1000
            }
        }
    }).toArray()
    console.log('天安门附近地铁站', r);

    client.close();
})()
```

案例:分页

封装以上api

新建models文件夹



config.js

```
module.exports = {
    url:'mongodb://localhost:27017',
    dbName:'classes',//数据库名字
}
```

db.js

```
const conf = require('./config')
const MongoClient =
require('mongodb').MongoClient;
const EventEmitter = require('events');
class Mongodb{
    constructor(conf) {
        this.conf = conf;
        this.emitter = new EventEmitter();
        this.client = new MongoClient(conf.url,
{useNewUrlParser:true});
        this.client.connect(err=>{
            if(err) throw err;
            console.log('链接数据库成功');

            this.emitter.emit('connect')
        })
    }
    once(eventName,cb){
        this.emitter.once(eventName,cb);
    }
    // 获取集合方法
    col(colName,dbName = this.conf.dbName){
        return this.client.db(dbName).collection(colName);
    }
}
module.exports = new Mongodb(conf);
```

testData.js

```
const mongodb = require('./db');
mongodb.once('connect', async () => {
    const col = mongodb.col('students');
    // 插入测试数据
    try {
        // 删除已存在的
        await col.deleteMany();

        await col.insertMany([
            {
                "name": "张三",
                "age": 20,
                "score": 90,
                "class": 1
            },
            {
                "name": "李四",
                "age": 30,
                "score": 100,
                "class": 2
            },
            {
                "name": "王五",
                "age": 30,
                "score": 70,
                "class": 1
            },
            {
                "name": "赵六",
                "age": 18,
                "score": 60,
                "class": 3
            },
            {
                "name": "小马哥",
                "age": 18,
                "score": 80,
                "class": 1
            },
            {
                "name": "小马哥2",
                "age": 18,
                "score": 80,
                "class": 1
            },
            {
                "name": "小马哥3",
                "age": 18,
                "score": 80,
                "class": 1
            },
            {
                "name": "小马哥4",
                "age": 18,
                "score": 80,
                "class": 1
            },
            {
                "name": "小马哥5",
                "age": 18,
                "score": 80,
                "class": 1
            },
            {
                "name": "小马哥6",
                "age": 18,
                "score": 80,
                "class": 1
            }
        ]);
        console.log('测试数据插入成功');

    } catch (error) {
        console.log('测试数据插入失败', error);

    }
})


```

server.js 服务器文件

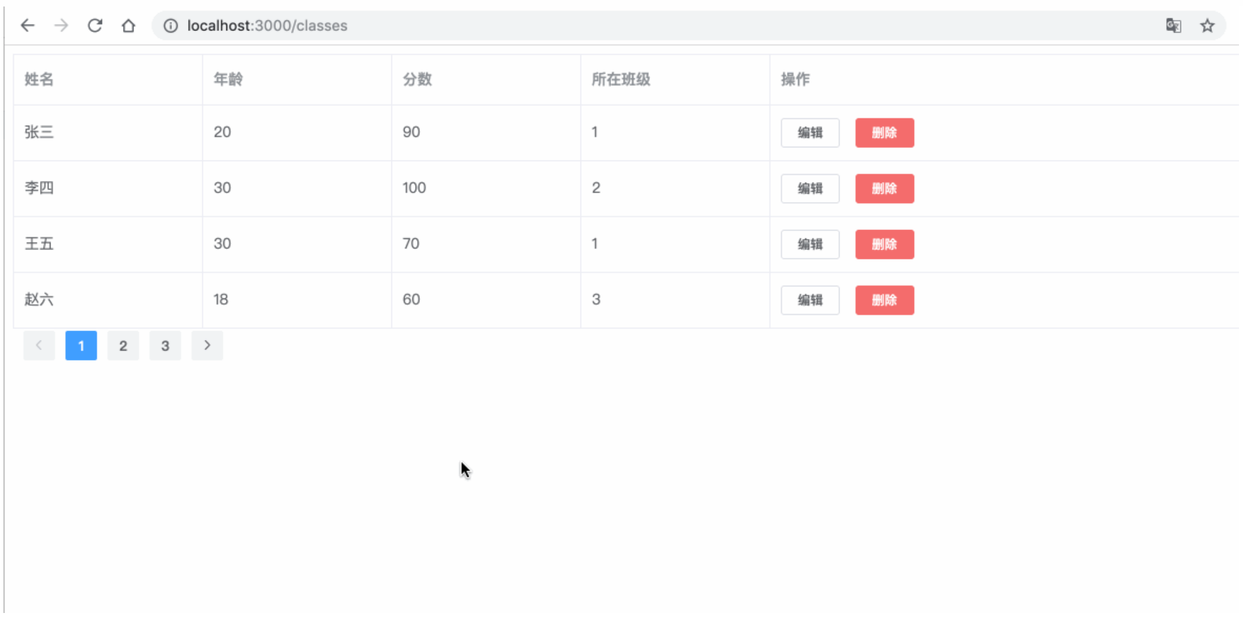
```
const mongo = require('./models/db');
const testData = require('./models/testdata');
const express = require('express');
const path = require('path');
const app = express();
//如果是post请求 需要拿到前端传来的参数,需要用body-parser进行解析
const bodyParser = require('body-parser');
app.use(bodyParser.json()); // for parsing application/json
app.use(bodyParser.urlencoded({ extended: true })); // for parsing application/x-www-form-urlencoded

app.get('/classes', (req, res) => {
    res.sendFile(path.resolve('./classes.html'))
})
app.get('/api/list', async (req, res) => {
    const pageSize = 4;
    const { page } = req.query;
    const students = mongo.col('students')
    const total = await students.find().count()
    const list = await students.find().skip((page - 1) * pageSize).limit(pageSize).toArray();
    // 查询总条数
    res.json({ ok: 1, data: { list, pagination: { total, page } } })

})

app.listen(3000, () => {
    console.log('监听3000端口成功');

});


```

clsses.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <!-- 引入样式 -->
    <link rel="stylesheet" href="https://unpkg.com/element-ui/lib/theme-chalk/index.css">
    <title>班级管理</title>
</head>

<body>
```

```html
    <div id="app">
        <el-table :data="classList" border style="width: 100%">
            <el-table-column prop="name" label="姓名" width="180">
            </el-table-column>
            <el-table-column prop="age" label="年龄" width="180">
            </el-table-column>
            <el-table-column prop="score" label="分数" width='180'>
            </el-table-column>
            <el-table-column prop="class" label="所在班级" width='180'>
            </el-table-column>
            <el-table-column label="操作">
                <template slot-scope="scope">
                    <el-button size="mini" @click="handleEdit(scope.$index, scope.row)">编辑</el-button>
                    <el-button size="mini" type="danger" @click="handleDelete(scope.$index, scope.row)">删除</el-button>
                </template>
            </el-table-column>
        </el-table>
        <el-pagination background layout="prev, pager, next" :total="total" @current-change="handleCurrentChange">
        </el-pagination>
    </div>

    <script src="https://cdn.jsdelivr.net/npm/vue@2.6.10/dist/vue.js"></script>
    <!-- 引入组件库 -->
    <script src="https://unpkg.com/element-ui/lib/index.js"></script>
    <script src="https://unpkg.com/axios/dist/axios.min.js"></script>
    <script>
        new Vue({
            el: '#app',
            data: {
                classList: [],
                total: 0,//页码总数
                page: 1 //默认页码为1
            },
            created() {
                this.getClassList()
            },
            methods: {
                handleEdit(index, row) {
                    console.log(index, row);
                },
                handleDelete(index, row) {
                    console.log(index, row);
                },
                handleCurrentChange(page) {
                    this.page = page;
                    this.getClassList();
                },
                async getClassList() {
                    try {
                        const {data} = await axios.get(`/api/list?page=${this.page}`).then(res => Promise.resolve(res
                            .data))
                        console.log(data);
                        this.classList = data.list;
                        this.total = Math.ceil(data.pagination.total / 4) * 10
                    } catch (error) {
                        console.log(error);
                    }
                }
            }
        })
    </script>
</body>

</html>
```

案例效果:


localhost:3000/classes

姓名	年龄	分数	所在班级	操作
张三	20	90	1	编辑 删除
李四	30	100	2	编辑 删除
王五	30	70	1	编辑 删除
赵六	18	60	3	编辑 删除

< 1 2 3 >

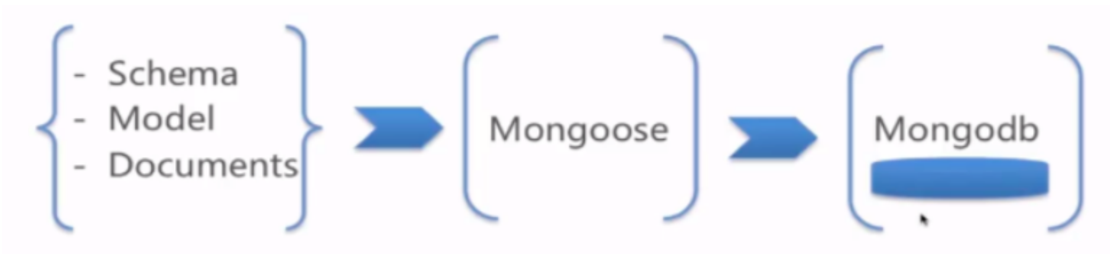
作业:

- 实现编辑,弹出模态框,数据修改
- 实现删除

ODM-Mongoose

基本使用

- 概述:优雅的NodeJS对象文档模型Object Document Model
- 两个特点
  - 通过关系型数据库的思想来设计非关系型数据库
  - 基于mongodb驱动,简化操作


- Schema
- Model
- Documents
Mongoose
Mongodb