nodejs链接mongodb的库mongoose

# 1. mongoose介绍

|  |
| --- |
|  |

## 我们把我们的练习代码结构调整一下，在根目录里面安装依赖

|  |
| --- |
|  |

## 然后我们安装mongoose：npm i mongoose

# 2.链接数据库

## 1最简单的测试代码

|  |
| --- |
| let mongoose = require('mongoose')  // mongoose.connect('mongodb://127.0.0.1/test')  mongoose.connect('mongodb://127.0.0.1/test1')    //设置数据库回调事件处理  var db = mongoose.connection;  db.on('error', console.error.bind(console, 'connection error:')); //连接失败  db.once('open', function() { //连接成功    // we're connected!    console.log("数据库连接成功");  });  db.on('close',()=>{    console.log("关闭数据库...");  }) |

### 效果：

|  |
| --- |
|  |

### 注意，即使需要连接的数据库不存在也会连接成功

### 响应关闭事件的实例代码

|  |
| --- |
|  |

### 补充说明：

### 1.关于strictQuery的设置，其实对我们以后的代码没有影响

|  |
| --- |
|  |

### 2.数据库回调处理‘open’事件，我们最好使用once来代替on

# 3.插入文档数据

## 插入一条数据

### 我们把上面的代码复制一份，改名02-mongoosbasic-insert.js，然后我们在这里编写插入数据的代码

|  |
| --- |
| //新增数据，使用新的promiser语法  let mongoose = require('mongoose')  mongoose.connect('mongodb://127.0.0.1/mybooks')    //设置数据库回调事件处理  var db = mongoose.connection;  db.on('error', console.error.bind(console, 'connection error:')); //连接失败  db.once('open', function() { //连接成功    // we're connected!    console.log("数据库连接成功");    //所有的业务代码都需要在这里写，不能写在外面    //1.新增数据    //1.1先创建文档的结构对象Schema    let bookSchema = new mongoose.Schema({      name:String,      author:String,      price:Number    })    //1.2创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查    let bookModel = mongoose.model("books",bookSchema)    //1.3新增数据 ，使用bookModel的create()方法  // bookModel.create({name:"天龙八部",author:"不详",price:18.8}).then(data=>{  //   console.log(data);  //   mongoose.disconnect()  // });  // bookModel.create({name:"倚天屠龙记",author:"金庸",price:18.8}).then(data=>{  //   console.log(data);  //   mongoose.disconnect()  // });  bookModel.create({name:"白蛇传",author:"未知",price:18.8}).then(data=>{    console.log(data);    mongoose.disconnect()   }).catch(err=>{     console.log(err);     return;   })  })  db.on('close',()=>{    console.log("关闭数据库...");  }) |

### 注意，新版本的mongoose的数据模型的create方法已经不接受回调函数了，这个和老师的不一样，可能那个时候是不一样的

#### 执行上面的代码，发现没有错误，我们在命令行窗口开启mongodb客户端，窗口一下发现有一个叫做bookshop的数据库里面有一个books集合，我们用找到我们新增的数据

|  |
| --- |
|  |

# 4.mongoose字段类型

|  |
| --- |
|  |

## 小案例，把02的代码复制一份改名：03-mongoosbasic-field-type.js

|  |
| --- |
| //字段类型，使用新的promise语法  let mongoose = require('mongoose')  mongoose.connect('mongodb://127.0.0.1/bookshop')    //设置数据库回调事件处理  var db = mongoose.connection;  db.on('error', console.error.bind(console, 'connection error:')); //连接失败  db.once('open', function() { //连接成功    // we're connected!    console.log("数据库连接成功");    //所有的业务代码都需要在这里写，不能写在外面    //1.新增数据    //1.1先创建文档的结构对象Schema,添加Boolean类型    let bookSchema = new mongoose.Schema({      name:String,      author:String,      price:Number,      is\_hot:Boolean,      tags:Array,      pub\_time:Date    })    //1.2创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查    let bookModel = mongoose.model("books",bookSchema)    //1.3新增数据 ，使用bookModel的create()方法  bookModel.create({       name:"西游记",       author:"吴承恩",       price:18.8,       is\_hot:true,       tags:['神话传说','四大名著'],       pub\_time:new Date()    }).then(data=>{      console.log(data);      mongoose.disconnect()    }).catch(err=>{      console.log(err);      return    })  })  db.on('close',()=>{    console.log("关闭数据库...");  }) |

### 效果如下

|  |
| --- |
|  |

### 注意：时间字段是比较重要的，建议加上时间字段，方便对特定时间的数据进行影响操作

### 虽然mongoose也可以使用Buffer字段来保存图片，视频的内容，但是我们一般包含把图片和视频内容直接保存到数据库里面，我们其实会把这些资源保存在静态资源文件夹里面，然后把他们的路径保存到数据库，使用的时候可以工具路径找到并且加载这些资源

## 关于Mixed字段的使用，必须使用类型mongoose的类型

|  |
| --- |
|  |

### 然后，它可以保存任何类型的值

## ObjectId类型的数据可以用来做外键，用于联合查询

# 5.mongoose字段验证

## 验证类型

|  |
| --- |
|  |
|  |
|  |

## 注意：

|  |
| --- |
|  |

## 内建验证器

|  |
| --- |
|  |
| **var** breakfastSchema = **new** Schema({  eggs: {  type: Number,  min: [6, 'Too few eggs'],  max: 12  },  bacon: {  type: Number,  required: [true, 'Why no bacon?']  },  drink: {  type: String,  enum: ['Coffee', 'Tea'],  required: **function**() {  **return** **this**.bacon > 3;  }  }  });  **var** Breakfast = db.model('Breakfast', breakfastSchema);  **var** badBreakfast = **new** Breakfast({  eggs: 2,  bacon: 0,  drink: 'Milk'  });  **var** error = badBreakfast.validateSync();  assert.equal(error.errors['eggs'].message,  'Too few eggs');  assert.ok(!error.errors['bacon']);  assert.equal(error.errors['drink'].message,  '`Milk` is not a valid enum value for path `drink`.');  badBreakfast.bacon = 5;  badBreakfast.drink = null;  error = badBreakfast.validateSync();  assert.equal(error.errors['drink'].message, 'Path `drink` is required.');  badBreakfast.bacon = null;  error = badBreakfast.validateSync();  assert.equal(error.errors['bacon'].message, 'Why no bacon?'); |

## 自定义验证器

如果内建检验器不够用了，你可以定义满足自己需要的检验器

自定义检验器通过传入一个检验函数来定义，更多细节请看 [SchemaType#validate() API 文档](https://mongoosejs.net/docs/api#schematype_SchemaType-validate)。

**var** userSchema = **new** Schema({

phone: {

type: String,

validate: {

validator: **function**(v) {

**return** /\d{3}-\d{3}-\d{4}/.test(v);

},

message: '{VALUE} is not a valid phone number!'

},

required: [true, 'User phone number required']

}

});

**var** User = db.model('user', userSchema);

**var** user = **new** User();

**var** error;

user.phone = '555.0123';

error = user.validateSync();

assert.equal(error.errors['phone'].message,

'555.0123 is not a valid phone number!');

user.phone = '';

error = user.validateSync();

assert.equal(error.errors['phone'].message,

'User phone number required');

user.phone = '201-555-0123';

*// Validation succeeds! Phone number is defined*

*// and fits `DDD-DDD-DDDD`*

error = user.validateSync();

assert.equal(error, null);

**[异步自定义验证器](https://mongoosejs.net/docs/validation" \l "%E5%BC%82%E6%AD%A5%E8%87%AA%E5%AE%9A%E4%B9%89%E9%AA%8C%E8%AF%81%E5%99%A8)**

自定义检验器可以是异步的。如果检验函数 返回 promise (像 async 函数)， mongoose 将会等待该 promise 完成。 如果你更喜欢使用回调函数，设置 isAsync 选项， mongoose 会将回调函数作为验证函数的第二个参数。

**var** userSchema = **new** Schema({

name: {

type: String,

*// You can also make a validator async by returning a promise. If you*

*// return a promise, do \*\*not\*\* specify the `isAsync` option.*

validate: **function**(v) {

**return** **new** Promise(**function**(resolve, reject) {

setTimeout(**function**() {

resolve(false);

}, 5);

});

}

},

phone: {

type: String,

validate: {

isAsync: true,

validator: **function**(v, cb) {

setTimeout(**function**() {

**var** phoneRegex = /\d{3}-\d{3}-\d{4}/;

**var** msg = v + ' is not a valid phone number!';

*// 第一个参数是布尔值，代表验证结果*

*// 第二个参数是报错信息*

cb(phoneRegex.test(v), msg);

}, 5);

},

*// 默认报错信息会被 `cb()` 第二个参数覆盖*

message: 'Default error message'

},

required: [true, 'User phone number required']

}

});

**var** User = db.model('User', userSchema);

**var** user = **new** User();

**var** error;

user.phone = '555.0123';

user.name = 'test';

user.validate(**function**(error) {

assert.ok(error);

assert.equal(error.errors['phone'].message,

'555.0123 is not a valid phone number!');

assert.equal(error.errors['name'].message,

'Validator failed for path `name` with value `test`');

});

**[验证错误](https://mongoosejs.net/docs/validation" \l "%E9%AA%8C%E8%AF%81%E9%94%99%E8%AF%AF)**

验证失败返回的 err 包含一个 ValidatorError 对象。 每一个 [ValidatorError](https://mongoosejs.net/docs/api#error-validation-js) 都有 kind、path、 value 和 message 属性。 ValidatorError 也可能有 reason 属性， 如果检验器抛出错误，这个属性会包含该错误原因。

**var** toySchema = **new** Schema({

color: String,

name: String

});

**var** validator = **function**(value) {

**return** /red|white|gold/i.test(value);

};

toySchema.path('color').validate(validator,

'Color `{VALUE}` not valid', 'Invalid color');

toySchema.path('name').validate(**function**(v) {

**if** (v !== 'Turbo Man') {

**throw** **new** Error('Need to get a Turbo Man for Christmas');

}

**return** true;

}, 'Name `{VALUE}` is not valid');

**var** Toy = db.model('Toy', toySchema);

**var** toy = **new** Toy({ color: 'Green', name: 'Power Ranger' });

toy.save(**function** (err) {

*// `err` is a ValidationError object*

*// `err.errors.color` is a ValidatorError object*

assert.equal(err.errors.color.message, 'Color `Green` not valid');

assert.equal(err.errors.color.kind, 'Invalid color');

assert.equal(err.errors.color.path, 'color');

assert.equal(err.errors.color.value, 'Green');

*// mongoose 5 新特性，如果验证器抛错，*

*// mongoose 会使用该错误信息。如果验证器返回 `false`，*

*// mongoose 会使用 'Name `Power Ranger` is not valid'。*

assert.equal(err.errors.name.message,

'Need to get a Turbo Man for Christmas');

assert.equal(err.errors.name.value, 'Power Ranger');

*// If your validator threw an error, the `reason` property will contain*

*// the original error thrown, including the original stack trace.*

assert.equal(err.errors.name.reason.message,

'Need to get a Turbo Man for Christmas');

assert.equal(err.name, 'ValidationError');

});

**[嵌套对象中的 Required 检验器](https://mongoosejs.net/docs/validation" \l "%E5%B5%8C%E5%A5%97%E5%AF%B9%E8%B1%A1%E4%B8%AD%E7%9A%84-required-%E6%A3%80%E9%AA%8C%E5%99%A8)**

定义嵌套对象的验证器需要特别注意。

**var** personSchema = **new** Schema({

name: {

first: String,

last: String

}

});

assert.throws(**function**() {

*// 这里会报错，因为 'name' 不是“完整成熟的路径”*

personSchema.path('name').required(true);

}, /Cannot.\*'required'/);

*// 要让嵌套对象 required，要使用单独的嵌套 schema*

**var** nameSchema = **new** Schema({

first: String,

last: String

});

personSchema = **new** Schema({

name: {

type: nameSchema,

required: true

}

});

**var** Person = db.model('Person', personSchema);

**var** person = **new** Person();

**var** error = person.validateSync();

assert.ok(error.errors['name']);

**[Update 验证器](https://mongoosejs.net/docs/validation" \l "update-%E9%AA%8C%E8%AF%81%E5%99%A8)**

上例中，你学习了 document 的验证。Mongoose 还支持验证 update() 和 findOneAndUpdate() 操作。 Update 验证器默认关闭，如需打开，请另外配置 runValidators。

注意：update 验证器默认关闭是因为里面有几个注意事项必须先了解。

**var** toySchema = **new** Schema({

color: String,

name: String

});

**var** Toy = db.model('Toys', toySchema);

Toy.schema.path('color').validate(**function** (value) {

**return** /blue|green|white|red|orange|periwinkle/i.test(value);

}, 'Invalid color');

**var** opts = { runValidators: true };

Toy.update({}, { color: 'bacon' }, opts, **function** (err) {

assert.equal(err.errors.color.message,

'Invalid color');

});

**[Update 验证器与 this](https://mongoosejs.net/docs/validation" \l "update-%E9%AA%8C%E8%AF%81%E5%99%A8%E4%B8%8E-this)**

update 验证器和 document 验证器有诸多不同。 上面的颜色验证函数，this 指向验证中的 document 。 然而 update 验证器运行时，被更新文档不一定存在于服务器内存， 所以 this 值未定义。

**var** toySchema = **new** Schema({

color: String,

name: String

});

toySchema.path('color').validate(**function**(value) {

*// When running in `validate()` or `validateSync()`, the*

*// validator can access the document using `this`.*

*// Does \*\*not\*\* work with update validators.*

**if** (**this**.name.toLowerCase().indexOf('red') !== -1) {

**return** value !== 'red';

}

**return** true;

});

**var** Toy = db.model('ActionFigure', toySchema);

**var** toy = **new** Toy({ color: 'red', name: 'Red Power Ranger' });

**var** error = toy.validateSync();

assert.ok(error.errors['color']);

**var** update = { color: 'red', name: 'Red Power Ranger' };

**var** opts = { runValidators: true };

Toy.update({}, update, opts, **function**(error) {

*// The update validator throws an error:*

*// "TypeError: Cannot read property 'toLowerCase' of undefined",*

*// because `this` is \*\*not\*\* the document being updated when using*

*// update validators*

assert.ok(error);

});

context 选项允许你把 update 验证器的 this 设置为 query。

toySchema.path('color').validate(**function**(value) {

*// When running update validators with the `context` option set to*

*// 'query', `this` refers to the query object.*

**if** (**this**.getUpdate().$set.name.toLowerCase().indexOf('red') !== -1) {

**return** value === 'red';

}

**return** true;

});

**var** Toy = db.model('Figure', toySchema);

**var** update = { color: 'blue', name: 'Red Power Ranger' };

*// Note the context option*

**var** opts = { runValidators: true, context: 'query' };

Toy.update({}, update, opts, **function**(error) {

assert.ok(error.errors['color']);

});

**[Update 验证器字段路径](https://mongoosejs.net/docs/validation" \l "update-%E9%AA%8C%E8%AF%81%E5%99%A8%E5%AD%97%E6%AE%B5%E8%B7%AF%E5%BE%84)**

另一个关键不同点是 update 验证器只运行于更新的字段。 下例中，因为 'name' 在更新操作未被指定，所以此次更新操作成功。

使用 update 验证器的时候， required 验证器**只会**在你对某个字段显式使用 $unset 才会触发。

**var** kittenSchema = **new** Schema({

name: { type: String, required: true },

age: Number

});

**var** Kitten = db.model('Kitten', kittenSchema);

**var** update = { color: 'blue' };

**var** opts = { runValidators: true };

Kitten.update({}, update, opts, **function**(err) {

*// 即使 'name' 没有指定也操作成功了*

});

**var** unset = { $unset: { name: 1 } };

Kitten.update({}, unset, opts, **function**(err) {

*// 'name' required， 操作失败*

assert.ok(err);

assert.ok(err.errors['name']);

});

**[Update 验证器只运行于指定字段路径](https://mongoosejs.net/docs/validation" \l "update-%E9%AA%8C%E8%AF%81%E5%99%A8%E5%8F%AA%E8%BF%90%E8%A1%8C%E4%BA%8E%E6%8C%87%E5%AE%9A%E5%AD%97%E6%AE%B5%E8%B7%AF%E5%BE%84)**

最后要注意的是：update 验证器**只**运行于下列更新操作：

* $set
* $unset
* $push (>= 4.8.0)
* $addToSet (>= 4.8.0)
* $pull (>= 4.12.0)
* $pullAll (>= 4.12.0)

例如，以下 update 成功执行，不管 number 的值，因为 update 验证器 无视 $inc 。同样， $push、$addToSet、 $pull 和 $pullAll 验证器 不会对数组自身验证，只会对数组中的元素验证。

**var** testSchema = **new** Schema({

number: { type: Number, max: 0 },

arr: [{ message: { type: String, maxlength: 10 } }]

});

*// Update 验证器不会作检查，所以你再仍然可以 `$push` 两个元素到数组*

*// 只要他们的 `message` 没有超长*

testSchema.path('arr').validate(**function**(v) {

**return** v.length < 2;

});

**var** Test = db.model('Test', testSchema);

**var** update = { $inc: { number: 1 } };

**var** opts = { runValidators: true };

Test.update({}, update, opts, **function**(error) {

*// 这里不会报错*

update = { $push: [{ message: 'hello' }, { message: 'world' }] };

Test.update({}, update, opts, **function**(error) {

*// 这里也不会报错*

});

});

**[$push 和 $addToSet](https://mongoosejs.net/docs/validation" \l "$push-%E5%92%8C-$addtoset)**

4.8.0 新特性： update 验证器也运行于 $push 和 $addToSet

**var** testSchema = **new** Schema({

numbers: [{ type: Number, max: 0 }],

docs: [{

name: { type: String, required: true }

}]

});

**var** Test = db.model('TestPush', testSchema);

**var** update = {

$push: {

numbers: 1,

docs: { name: null }

}

};

**var** opts = { runValidators: true };

Test.update({}, update, opts, **function**(error) {

assert.ok(error.errors['numbers']);

assert.ok(error.errors['docs']);

});

## 课堂案例

### 把03文件复制一份，改名04-mongoosbasic-field-verify.js，然后我们来做字段验证，我们把name改为必须字段，但是我们在新增数据时候故意把name字段注释掉

|  |
| --- |
| //字段验证，使用promise写法  let mongoose = require('mongoose')  mongoose.connect('mongodb://127.0.0.1/bookshop')    //设置数据库回调事件处理  var db = mongoose.connection;  db.on('error', console.error.bind(console, 'connection error:')); //连接失败  db.once('open', function() { //连接成功    // we're connected!    console.log("数据库连接成功");    //所有的业务代码都需要在这里写，不能写在外面    //1.新增数据    //1.1先创建文档的结构对象Schema,添加Boolean类型    let bookSchema = new mongoose.Schema({      name:{        type:String,        required:true,        unique:true      },      author:{        type:String,        default:'匿名'      },      style:{        type:String,        enum:['言情','推理','小说','城市','战争']      },      price:Number,      is\_hot:Boolean,      tags:Array,      pub\_time:Date    })    //1.2创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查    let bookModel = mongoose.model("books",bookSchema)    //1.3新增数据 ，使用bookModel的create()方法  bookModel.create({       //name:"红楼梦",       author:"曹雪芹",       style:'言情',       price:18.8,       is\_hot:true,       tags:['神话传说','四大名著'],       pub\_time:new Date()   }).then(data=>{    console.log(data);    mongoose.disconnect()   }).catch(err=>{      console.log(err);      return;   })  })  db.on('close',()=>{    console.log("关闭数据库...");  }) |

### 效果：此时会有下面的报错

|  |
| --- |
|  |

### 说明我们的验证是有效的

### 我们把name取消注释，此时它又正常了，现在我们来设置author字段的默认值，然后我们故意不传递author字段

|  |
| --- |
| //字段验证，使用promise写法  let mongoose = require('mongoose')  mongoose.connect('mongodb://127.0.0.1/bookshop')    //设置数据库回调事件处理  var db = mongoose.connection;  db.on('error', console.error.bind(console, 'connection error:')); //连接失败  db.once('open', function() { //连接成功    // we're connected!    console.log("数据库连接成功");    //所有的业务代码都需要在这里写，不能写在外面    //1.新增数据    //1.1先创建文档的结构对象Schema,添加Boolean类型    let bookSchema = new mongoose.Schema({      name:{        type:String,        required:true,        unique:true      },      author:{        type:String,        default:'匿名'      },      style:{        type:String,        enum:['言情','推理','小说','城市','战争']      },      price:Number,      is\_hot:Boolean,      tags:Array,      pub\_time:Date    })    //1.2创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查    let bookModel = mongoose.model("books",bookSchema)    //1.3新增数据 ，使用bookModel的create()方法  bookModel.create({       name:"红楼梦",       //author:"曹雪芹",       style:'言情',       price:18.8,       is\_hot:true,       tags:['神话传说','四大名著'],       pub\_time:new Date()   }).then(data=>{    console.log(data);    mongoose.disconnect()   }).catch(err=>{      console.log(err);      return;   })  })  db.on('close',()=>{    console.log("关闭数据库...");  }) |

### 效果

|  |
| --- |
|  |

### 我们来看看enum，我们把author字段还原回来，然后给文档结构添加一个style字段，类型是一个enum，输入的时候就会又提示

|  |
| --- |
|  |

### 我们完成内容输入

//字段验证，使用promise写法

let mongoose = require('mongoose')

mongoose.connect('mongodb://127.0.0.1/bookshop')

//设置数据库回调事件处理

var db = mongoose.connection;

db.on('error', console.error.bind(console, 'connection error:')); //连接失败

db.once('open', function() { //连接成功

  // we're connected!

  console.log("数据库连接成功");

  //所有的业务代码都需要在这里写，不能写在外面

  //1.新增数据

  //1.1先创建文档的结构对象Schema,添加Boolean类型

  let bookSchema = new mongoose.Schema({

    name:{

      type:String,

      required:true,

      unique:true

    },

    author:{

      type:String,

      default:'匿名'

    },

    style:{

      type:String,

      enum:['言情','推理','小说','城市','战争']

    },

    price:Number,

    is\_hot:Boolean,

    tags:Array,

    pub\_time:Date

  })

  //1.2创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查

  let bookModel = mongoose.model("books",bookSchema)

  //1.3新增数据 ，使用bookModel的create()方法

bookModel.create({

     name:"红楼梦",

     author:"曹雪芹",

     style:'言情',

     price:18.8,

     is\_hot:true,

     tags:['神话传说','四大名著'],

     pub\_time:new Date()

 }).then(data=>{

  console.log(data);

  mongoose.disconnect()

 }).catch(err=>{

    console.log(err);

    return;

 })

})

db.on('close',()=>{

  console.log("关闭数据库...");

})

### 效果如下：

|  |
| --- |
|  |

### 注意，如果你输入的不是这些值的其中一个，就会报错

|  |
| --- |
|  |

### 还有一个unique属性，需要新建的集合才有用，在旧集合没有效果

|  |
| --- |
|  |

### 然后在程序中插入同一条数据2次，就会出现下面的错误

|  |
| --- |
|  |

### 我们也来试一试，我们把name字段的unique属性设置为true

|  |
| --- |
| //字段验证，使用promise写法  let mongoose = require('mongoose')  mongoose.connect('mongodb://127.0.0.1/bookshop')    //设置数据库回调事件处理  var db = mongoose.connection;  db.on('error', console.error.bind(console, 'connection error:')); //连接失败  db.once('open', function() { //连接成功    // we're connected!    console.log("数据库连接成功");    //所有的业务代码都需要在这里写，不能写在外面    //1.新增数据    //1.1先创建文档的结构对象Schema,添加Boolean类型    let bookSchema = new mongoose.Schema({      name:{        type:String,        required:true,        unique:true      },      author:{        type:String,        default:'匿名'      },      style:{        type:String,        enum:['言情','推理','小说','城市','战争']      },      price:Number,      is\_hot:Boolean,      tags:Array,      pub\_time:Date    })    //1.2创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查    let bookModel = mongoose.model("books",bookSchema)    //1.3新增数据 ，使用bookModel的create()方法  bookModel.create({       name:"红楼梦",       author:"曹雪芹",       style:'言情',       price:18.8,       is\_hot:true,       tags:['神话传说','四大名著'],       pub\_time:new Date()   }).then(data=>{    console.log(data);    mongoose.disconnect()   }).catch(err=>{      console.log(err);      return;   })  })  db.on('close',()=>{    console.log("关闭数据库...");  }) |

### 然后我们把books集合删除

|  |
| --- |
|  |

### 然后我们来运行程序，第一次，成功新建了数据

|  |
| --- |
|  |
|  |

### 当我们再一次新增一模一样的数据，就会报一个错误说key重复了

|  |
| --- |
|  |

### 说明这个设置起作用了，这个属性其实很有用，如防止网站添加同名用户以及其他不能重复的场合等等

## 注意：新版本的mongoose的数据模型的create方法取消了回调作为参数，但是他返回的是一个promise，我们可以在then函数里面输出我们插入的数据，还可以做一些处理，我把02文件的代码修改了一下，如下：

### 02-mongoosbasic-insert.js

|  |
| --- |
| let mongoose = require('mongoose')  mongoose.connect('mongodb://127.0.0.1/mybooks')    //设置数据库回调事件处理  var db = mongoose.connection;  db.on('error', console.error.bind(console, 'connection error:')); //连接失败  db.once('open', function() { //连接成功    // we're connected!    console.log("数据库连接成功");    //所有的业务代码都需要在这里写，不能写在外面    //1.新增数据    //1.1先创建文档的结构对象Schema    let bookSchema = new mongoose.Schema({      name:String,      author:String,      price:Number    })    //1.2创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查    let bookModel = mongoose.model("books",bookSchema)    //1.3新增数据 ，使用bookModel的create()方法  bookModel.create({name:"天龙八部",author:"不详",price:18.8}).then(data=>{    console.log(data);    mongoose.disconnect()  });  })  db.on('close',()=>{    console.log("关闭数据库...");  }) |

## 效果

|  |
| --- |
|  |

## 补充：mongoose的模型对象也有插入多条数据的功能，我们创建一个00-insermany.js文件来实现这个功能，操作的是bookshop数据库

|  |
| --- |
| //插入多条文档数据,promise语法  let mongoose = require('mongoose')  mongoose.connect('mongodb://127.0.0.1/bookshop')    //设置数据库回调事件处理  var db = mongoose.connection;  db.on('error', console.error.bind(console, 'connection error:')); //连接失败  db.once('open', function() { //连接成功    // we're connected!    console.log("数据库连接成功");    //所有的业务代码都需要在这里写，不能写在外面    //1.新增数据    //1.1先创建文档的结构对象Schema,添加Boolean类型    let bookSchema = new mongoose.Schema({      name:{        type:String,        required:true,        unique:true      },      author:{        type:String,        default:'匿名'      },      style:{        type:String,        enum:['言情','推理','小说','城市','战争']      },      price:Number,      is\_hot:Boolean,      tags:Array,      pub\_time:Date    })    //1.2创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查    let bookModel = mongoose.model("books",bookSchema)    //1.3新增多条数据 ，使用bookModel的insertMany()方法    let newBooks = [      {          name:"天龙八部",          // author:"",          style:'言情',          price:18.8,          is\_hot:true,          tags:['神话传说','武侠小说'],          pub\_time:new Date()     },      {          name:"射雕英雄传",          // author:"曹雪芹",          style:'言情',          price:18.8,          is\_hot:true,          tags:['神话传说','四大名著'],          pub\_time:new Date()     },      {          name:"金瓶梅",          // author:"曹雪芹",          style:'言情',          price:18.8,          is\_hot:true,          tags:['神话传说','四大禁书'],          pub\_time:new Date()     },      {          name:"倚天屠龙记",          author:"曹雪芹",          style:'言情',          price:18.8,          is\_hot:true,          tags:['神话传说','武侠小说'],          pub\_time:new Date()     },    ]    bookModel.insertMany(newBooks).then(data=>{      console.log(data);      mongoose.disconnect()    })  })  db.on('close',()=>{    console.log("关闭数据库...");  }) |

### 我们运行一下，插入多条数据，效果如下

|  |
| --- |
| 零基础学习笔记\nodejs-code\learnmongodb\mongoosedemo> node .\00-insermany.js  数据库连接成功  [  {  name: '天龙八部',  author: '匿名',  style: '言情',  price: 18.8,  is\_hot: true,  tags: [ '神话传说', '武侠小说' ],  pub\_time: 2024-02-28T23:06:22.534Z,  \_id: new ObjectId('65dfbc6e10b2936a63de6201'),  \_\_v: 0  },  {  name: '射雕英雄传',  author: '匿名',  style: '言情',  price: 18.8,  is\_hot: true,  tags: [ '神话传说', '四大名著' ],  pub\_time: 2024-02-28T23:06:22.535Z,  \_id: new ObjectId('65dfbc6e10b2936a63de6202'),  \_\_v: 0  },  {  name: '金瓶梅',  author: '匿名',  style: '言情',  price: 18.8,  is\_hot: true,  tags: [ '神话传说', '四大禁书' ],  pub\_time: 2024-02-28T23:06:22.535Z,  \_id: new ObjectId('65dfbc6e10b2936a63de6203'),  \_\_v: 0  },  {  name: '倚天屠龙记',  author: '曹雪芹',  style: '言情',  price: 18.8,  is\_hot: true,  tags: [ '神话传说', '武侠小说' ],  pub\_time: 2024-02-28T23:06:22.535Z,  \_id: new ObjectId('65dfbc6e10b2936a63de6204'),  \_\_v: 0  }  ]  关闭数据库... |

### 插入多条数据成功。用命令行查看

|  |
| --- |
|  |

6、

# 6、更新文档数据

## 1）更新一条数据

### 创建一个05-mongoosbasic-update-doc.js文件，添加修改一条数据的代码如把三国演义这本书的价钱改为19.8

|  |
| --- |
| //修改数据，使用promise新语法  let mongoose = require('mongoose')  mongoose.connect('mongodb://127.0.0.1/bookshop')    //设置数据库回调事件处理  var db = mongoose.connection;  db.on('error', console.error.bind(console, 'connection error:')); //连接失败  db.once('open', function() { //连接成功    // we're connected!    console.log("数据库连接成功");    //所有的业务代码都需要在这里写，不能写在外面    //1.新增数据    //1.1先创建文档的结构对象Schema,添加Boolean类型    let bookSchema = new mongoose.Schema({      name:{        type:String,        required:true,        unique:true      },      author:{        type:String,        default:'匿名'      },      style:{        type:String,        enum:['言情','推理','小说','城市','战争']      },      price:Number,      is\_hot:Boolean,      tags:Array,      pub\_time:Date    })    //1.2创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查    let bookModel = mongoose.model("books",bookSchema)    //1.3新增数据 ，使用bookModel的create()方法  // bookModel.create({  //      name:"红楼梦",  //      author:"曹雪芹",  //      style:'言情',  //      price:18.8,  //      is\_hot:true,  //      tags:['神话传说','四大名著'],  //      pub\_time:new Date()  // }).then(data=>{  //   console.log(data);  //   mongoose.disconnect()  // });  ;  //更新数据  // bookModel.updateOne({name:"水浒传"},{$set:{price:20.5}})  //       .then(data=>{  //         console.log(data);  //         mongoose.disconnect()  //       })  bookModel.updateOne({name:"三国演义"},{$set:{price:19.8}})        .then(data=>{          console.log(data);          mongoose.disconnect()   })  })  db.on('close',()=>{    console.log("关闭数据库...");  }) |

### 效果：

|  |
| --- |
|  |

### 用命令行查看，发现的确是修改了

|  |
| --- |
|  |

### 我们用同样的方法把倚天屠龙记改为22.7

|  |
| --- |
|  |

## 2）更新多条数据：我们写一个方法把剩下的4条18.8的文档记录的价钱改为17.8

|  |
| --- |
| //修改数据，使用promise新语法  let mongoose = require('mongoose')  mongoose.connect('mongodb://127.0.0.1/bookshop')    //设置数据库回调事件处理  var db = mongoose.connection;  db.on('error', console.error.bind(console, 'connection error:')); //连接失败  db.once('open', function() { //连接成功    // we're connected!    console.log("数据库连接成功");    //所有的业务代码都需要在这里写，不能写在外面    //1.新增数据    //1.1先创建文档的结构对象Schema,添加Boolean类型    let bookSchema = new mongoose.Schema({      name:{        type:String,        required:true,        unique:true      },      author:{        type:String,        default:'匿名'      },      style:{        type:String,        enum:['言情','推理','小说','城市','战争']      },      price:Number,      is\_hot:Boolean,      tags:Array,      pub\_time:Date    })    //1.2创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查    let bookModel = mongoose.model("books",bookSchema)    //1.3新增数据 ，使用bookModel的create()方法  // bookModel.create({  //      name:"红楼梦",  //      author:"曹雪芹",  //      style:'言情',  //      price:18.8,  //      is\_hot:true,  //      tags:['神话传说','四大名著'],  //      pub\_time:new Date()  // }).then(data=>{  //   console.log(data);  //   mongoose.disconnect()  // });  ;  //更新数据  //1.更新一条数据  // bookModel.updateOne({name:"水浒传"},{$set:{price:20.5}})  //       .then(data=>{  //         console.log(data);  //         mongoose.disconnect()  //       })  // bookModel.updateOne({name:"倚天屠龙记"},{$set:{price:22.7}})  //       .then(data=>{  //         console.log(data);  //         mongoose.disconnect()  //       })  //2.更新多条数据  bookModel.updateMany({price:18.8},{$set:{price:17.8}})        .then(data=>{          console.log(data);          mongoose.disconnect()  })  })  db.on('close',()=>{    console.log("关闭数据库...");  }) |

### 效果：

|  |
| --- |
|  |

### 我们到命令行窗口窗口，效果如下

|  |
| --- |
|  |

### 注意：有时候我们需要一点自学能力，因为老师讲课的那一段时间代码的写法和现在可能不一样了，需要多查看文档，多多练习，然后比较的顺序和老师的有点不一样也没有关系，只要能学习到东西就是好的

### 其实mongoose修改文档数据还可以这么写，它不会覆盖，而是只更新指定的字段，其他字段不变

|  |
| --- |
| //修改数据，使用promise新语法  let mongoose = require('mongoose')  mongoose.connect('mongodb://127.0.0.1/bookshop2')    //设置数据库回调事件处理  var db = mongoose.connection;  db.on('error', console.error.bind(console, 'connection error:')); //连接失败  db.once('open', function() { //连接成功    // we're connected!    console.log("数据库连接成功");    //所有的业务代码都需要在这里写，不能写在外面    //1.新增数据    //1.1先创建文档的结构对象Schema,添加Boolean类型    let bookSchema = new mongoose.Schema({      name:{        type:String,        required:true,        unique:true      },      author:{        type:String,        default:'匿名'      },      style:{        type:String,        enum:['言情','推理','小说','城市','战争']      },      price:Number,      is\_hot:Boolean,      tags:Array,      pub\_time:Date    })    //1.2创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查    let bookModel = mongoose.model("books",bookSchema)    //1.3新增数据 ，使用bookModel的create()方法  // bookModel.create({  //      name:"红楼梦",  //      author:"曹雪芹",  //      style:'言情',  //      price:18.8,  //      is\_hot:true,  //      tags:['神话传说','四大名著'],  //      pub\_time:new Date()  // }).then(data=>{  //   console.log(data);  //   mongoose.disconnect()  // });  ;  //更新数据  //1.更新一条数据  // bookModel.updateOne({name:"水浒传"},{$set:{price:20.5}})  //       .then(data=>{  //         console.log(data);  //         mongoose.disconnect()  //       })  // 还可以这么写  // bookModel.updateOne({name:"水浒传"},{price:20.5}) //这里和命令行不一样，它只会更新指定的字段而不是覆盖  //       .then(data=>{  //         console.log(data);  //         mongoose.disconnect()  //       })  //2.更新多条数据  bookModel.updateMany({price:17.8},{price:18.8})        .then(data=>{          console.log(data);          mongoose.disconnect()       }).catch(err=>{        console.log(err);        return    })  })  db.on('close',()=>{    console.log("关闭数据库...");  }) |

# 7.删除文档数据（相当于关系数据库的记录）

## 在删除之前，我们先新建几条数据，新增后数据如下

|  |
| --- |
| > db.books.find();  { "\_id" : ObjectId("65dfac18d08ecefefbdaddbc"), "name" : "西游记", "author" : "吴承恩", "style" : "言情", "price" : 18.8, "is\_hot" : true, "tags" : [ "神话传说", "四大名著" ], "pub\_time" : ISODate("2024-02-28T21:56:40.416Z"), "\_\_v" : 0 }  { "\_id" : ObjectId("65dfafcb77c245bb92c6749c"), "name" : "水浒传", "author" : "施耐庵", "style" : "城市", "price" : 18.8, "is\_hot" : true, "tags" : [ "神话传说", "四大名著" ], "pub\_time" : ISODate("2024-02-28T22:12:27.331Z"), "\_\_v" : 0 }  { "\_id" : ObjectId("65dfb023f37501f1bf2f5f6d"), "name" : "三国演义", "author" : "罗贯中", "style" : "战争", "price" : 18.8, "is\_hot" : true, "tags" : [ "神话传说", "四大名著" ], "pub\_time" : ISODate("2024-02-28T22:13:55.453Z"), "\_\_v" : 0 }  { "\_id" : ObjectId("65dfb04de75e93754aa17e97"), "name" : "红楼梦", "author" : "曹雪芹", "style" : "言情", "price" : 18.8, "is\_hot" : true, "tags" : [ "神话传说", "四大名著" ], "pub\_time" : ISODate("2024-02-28T22:14:37.989Z"), "\_\_v" : 0 }  > |

## 1）删除一条数据

## 我们把04代码复制一份改名06-mongoosbasic-delete-doc.js

|  |
| --- |
| //删除数据，使用promise新语法  let mongoose = require('mongoose')  mongoose.connect('mongodb://127.0.0.1/bookshop')    //设置数据库回调事件处理  var db = mongoose.connection;  db.on('error', console.error.bind(console, 'connection error:')); //连接失败  db.once('open', function() { //连接成功    // we're connected!    console.log("数据库连接成功");    //所有的业务代码都需要在这里写，不能写在外面    //1.新增数据    //1.1先创建文档的结构对象Schema,添加Boolean类型    let bookSchema = new mongoose.Schema({      name:{        type:String,        required:true,        unique:true      },      author:{        type:String,        default:'匿名'      },      style:{        type:String,        enum:['言情','推理','小说','城市','战争']      },      price:Number,      is\_hot:Boolean,      tags:Array,      pub\_time:Date    })    //1.2创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查    let bookModel = mongoose.model("books",bookSchema)    //1.3新增数据 ，使用bookModel的create()方法  // bookModel.create({  //      name:"红楼梦",  //      author:"曹雪芹",  //      style:'言情',  //      price:18.8,  //      is\_hot:true,  //      tags:['神话传说','四大名著'],  //      pub\_time:new Date()  // }).then(data=>{  //   console.log(data);  //   mongoose.disconnect()  // });  ;  // 2.删除数据  //2.1删除一条数据  bookModel.deleteOne({    \_id:"65dfac18d08ecefefbdaddbc"  }).then(data=>{    console.log(data);    mongoose.disconnect() //实际开发中，不能机场6断开数据库连接，这里是为了方便调试  })  //2.2删除多条数据  })  db.on('close',()=>{    console.log("关闭数据库...");  }) |

### 效果：

|  |
| --- |
|  |

## 可以看到西游记被删除了

|  |
| --- |
|  |

## 2）删除多条数据，例如我们把数据库bookshop里面所有17.8元的书都删除

### 06-mongoosbasic-delete-doc.js

|  |
| --- |
| //删除数据，使用promise新语法  let mongoose = require('mongoose')  mongoose.connect('mongodb://127.0.0.1/bookshop')    //设置数据库回调事件处理  var db = mongoose.connection;  db.on('error', console.error.bind(console, 'connection error:')); //连接失败  db.once('open', function() { //连接成功    // we're connected!    console.log("数据库连接成功");    //所有的业务代码都需要在这里写，不能写在外面    //1.新增数据    //1.1先创建文档的结构对象Schema,添加Boolean类型    let bookSchema = new mongoose.Schema({      name:{        type:String,        required:true,        unique:true      },      author:{        type:String,        default:'匿名'      },      style:{        type:String,        enum:['言情','推理','小说','城市','战争']      },      price:Number,      is\_hot:Boolean,      tags:Array,      pub\_time:Date    })    //1.2创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查    let bookModel = mongoose.model("books",bookSchema)    //1.3新增数据 ，使用bookModel的create()方法  // 2.删除数据  //2.1删除一条数据    // bookModel.deleteOne({    //   \_id:"65dfac18d08ecefefbdaddbc"    // }).then(data=>{    //   console.log(data);    //   mongoose.disconnect() //实际开发中，不能机场6断开数据库连接，这里是为了方便调试    // }).catch(err=>{    //   console.log(err);    //   return    // })  //2.2删除多条数据,  //例如我们把数据库里面所有17.8元是书都删除，为了方便数据恢复，我把bookshop数据库复制了一份改名bookshop2      bookModel.deleteMany({        price:17.8      }).then(data=>{        console.log(data);        mongoose.disconnect() //实际开发中，不能机场6断开数据库连接，这里是为了方便调试      }).catch(err=>{        console.log(err);        return      })  })  db.on('close',()=>{    console.log("关闭数据库...");  }) |

### 运行成功

|  |
| --- |
|  |

### 然后我们从命令行窗口查看

|  |
| --- |
|  |

### 我们把所有17.8元的书都删除了

# 8.读取文档数据（查找数据），为了更方便练习，我们在命令行先删除bookshop数据库，然后把bookshop2复制一份到bookshop

|  |
| --- |
|  |

## 我们新建一个07-mongoosbasic-find-doc.js文件，在里面写查询代码

## 1）查询一个

|  |
| --- |
| //查找数据，使用promise新语法  let mongoose = require('mongoose')  mongoose.connect('mongodb://127.0.0.1/bookshop')    //设置数据库回调事件处理  var db = mongoose.connection;  db.on('error', console.error.bind(console, 'connection error:')); //连接失败  db.once('open', function() { //连接成功    // we're connected!    console.log("数据库连接成功");    //所有的业务代码都需要在这里写，不能写在外面    //1.新增数据    //1.1先创建文档的结构对象Schema,添加Boolean类型    let bookSchema = new mongoose.Schema({      name:{        type:String,        required:true,        unique:true      },      author:{        type:String,        default:'匿名'      },      style:{        type:String,        enum:['言情','推理','小说','城市','战争']      },      price:Number,      is\_hot:Boolean,      tags:Array,      pub\_time:Date    })    //1.2创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查    let bookModel = mongoose.model("books",bookSchema)    //1.3新增数据 ，使用bookModel的create()方法    //查找一条数据，我们先在命令行把bookshop2的数据重新拷贝到bookshop，需要先删除bookshop数据库    bookModel.findOne({name:"红楼梦"})        .then(data=>{          console.log(data);          mongoose.disconnect()        })        .catch(err=>{          console.log(err);          return        })  })  db.on('close',()=>{    console.log("关闭数据库...");  }) |

### 结果：

|  |
| --- |
|  |

## 2)根据id来查询数据

|  |
| --- |
| //查找数据，使用promise新语法  let mongoose = require('mongoose')  mongoose.connect('mongodb://127.0.0.1/bookshop')    //设置数据库回调事件处理  var db = mongoose.connection;  db.on('error', console.error.bind(console, 'connection error:')); //连接失败  db.once('open', function() { //连接成功    // we're connected!    console.log("数据库连接成功");    //所有的业务代码都需要在这里写，不能写在外面      //先创建文档的结构对象Schema,添加Boolean类型    let bookSchema = new mongoose.Schema({      name:{        type:String,        required:true,        unique:true      },      author:{        type:String,        default:'匿名'      },      style:{        type:String,        enum:['言情','推理','小说','城市','战争']      },      price:Number,      is\_hot:Boolean,      tags:Array,      pub\_time:Date    })    //创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查    let bookModel = mongoose.model("books",bookSchema)    //新增数据 ，使用bookModel的create()方法    //查找一条数据，我们先在命令行把bookshop2的数据重新拷贝到bookshop，需要先删除bookshop数据库    // bookModel.findOne({name:"红楼梦"})    //     .then(data=>{    //       console.log(data);    //       mongoose.disconnect()    //     })    //     .catch(err=>{    //       console.log(err);    //       return    //   })    //根据id来查找     bookModel.findById({\_id:'65dfafcb77c245bb92c6749c'})        .then(data=>{          console.log(data);          mongoose.disconnect()        })        .catch(err=>{          console.log(err);          return        })    })  db.on('close',()=>{    console.log("关闭数据库...");  }) |

### 结果

|  |
| --- |
|  |

## 还可以这么写

|  |
| --- |
| //查找数据，使用promise新语法  let mongoose = require('mongoose')  mongoose.connect('mongodb://127.0.0.1/bookshop')    //设置数据库回调事件处理  var db = mongoose.connection;  db.on('error', console.error.bind(console, 'connection error:')); //连接失败  db.once('open', function() { //连接成功    // we're connected!    console.log("数据库连接成功");    //所有的业务代码都需要在这里写，不能写在外面      //先创建文档的结构对象Schema,添加Boolean类型    let bookSchema = new mongoose.Schema({      name:{        type:String,        required:true,        unique:true      },      author:{        type:String,        default:'匿名'      },      style:{        type:String,        enum:['言情','推理','小说','城市','战争']      },      price:Number,      is\_hot:Boolean,      tags:Array,      pub\_time:Date    })    //创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查    let bookModel = mongoose.model("books",bookSchema)    //新增数据 ，使用bookModel的create()方法    //查找一条数据，我们先在命令行把bookshop2的数据重新拷贝到bookshop，需要先删除bookshop数据库    // bookModel.findOne({name:"红楼梦"})    //     .then(data=>{    //       console.log(data);    //       mongoose.disconnect()    //     })    //     .catch(err=>{    //       console.log(err);    //       return    //   })    //根据id来查找     bookModel.findById('65dfbc6e10b2936a63de6204')        .then(data=>{          console.log(data);          mongoose.disconnect()        })        .catch(err=>{          console.log(err);          return        })    })  db.on('close',()=>{    console.log("关闭数据库...");  }) |

### 结果

|  |
| --- |
|  |

## 3)查询多个文档数据，例如查找所有18.8元的书，使用find()方法

|  |
| --- |
| //查找数据，使用promise新语法  let mongoose = require('mongoose')  mongoose.connect('mongodb://127.0.0.1/bookshop')    //设置数据库回调事件处理  var db = mongoose.connection;  db.on('error', console.error.bind(console, 'connection error:')); //连接失败  db.once('open', function() { //连接成功    // we're connected!    console.log("数据库连接成功");    //所有的业务代码都需要在这里写，不能写在外面    //1.新增数据    //1.1先创建文档的结构对象Schema,添加Boolean类型    let bookSchema = new mongoose.Schema({      name:{        type:String,        required:true,        unique:true      },      author:{        type:String,        default:'匿名'      },      style:{        type:String,        enum:['言情','推理','小说','城市','战争']      },      price:Number,      is\_hot:Boolean,      tags:Array,      pub\_time:Date    })    //1.2创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查    let bookModel = mongoose.model("books",bookSchema)    //1.3新增数据 ，使用bookModel的create()方法    //查找一条数据，我们先在命令行把bookshop2的数据重新拷贝到bookshop，需要先删除bookshop数据库    // bookModel.findOne({name:"红楼梦"})    //     .then(data=>{    //       console.log(data);    //       mongoose.disconnect()    //     })    //     .catch(err=>{    //       console.log(err);    //       return    //   })    //查询多条数据，例如查找所有18.8元的书    bookModel.find({price:18.8})    .then(data=>{      console.log(data);      mongoose.disconnect()    })    .catch(err=>{      console.log(err);      return   })  })  db.on('close',()=>{    console.log("关闭数据库...");  }) |

### 效果：

|  |
| --- |
| 数据库连接成功  [  {  \_id: new ObjectId('65dfb04de75e93754aa17e97'),  name: '红楼梦',  author: '曹雪芹',  style: '言情',  price: 18.8,  is\_hot: true,  tags: [ '神话传说', '四大名著' ],  pub\_time: 2024-02-28T22:14:37.989Z,  \_\_v: 0  },  {  \_id: new ObjectId('65dfbc6e10b2936a63de6201'),  name: '天龙八部',  author: '匿名',  style: '言情',  price: 18.8,  is\_hot: true,  tags: [ '神话传说', '武侠小说' ],  pub\_time: 2024-02-28T23:06:22.534Z,  \_\_v: 0  },  {  \_id: new ObjectId('65dfbc6e10b2936a63de6202'),  name: '射雕英雄传',  author: '匿名',  style: '言情',  price: 18.8,  is\_hot: true,  tags: [ '神话传说', '四大名著' ],  pub\_time: 2024-02-28T23:06:22.535Z,  \_\_v: 0  },  {  \_id: new ObjectId('65dfbc6e10b2936a63de6203'),  name: '金瓶梅',  author: '匿名',  style: '言情',  price: 18.8,  is\_hot: true,  tags: [ '神话传说', '四大禁书' ],  pub\_time: 2024-02-28T23:06:22.535Z,  \_\_v: 0  }  ]  关闭数据库... |

### 需要特别注意，插入，更新和删除都有many版本，但是查找没有many版本，找一个是findOne({}),找多个是find({})

### 查询所有数据

|  |
| --- |
| //查找数据，使用promise新语法  let mongoose = require('mongoose')  mongoose.connect('mongodb://127.0.0.1/bookshop')    //设置数据库回调事件处理  var db = mongoose.connection;  db.on('error', console.error.bind(console, 'connection error:')); //连接失败  db.once('open', function() { //连接成功    // we're connected!    console.log("数据库连接成功");    //所有的业务代码都需要在这里写，不能写在外面      //先创建文档的结构对象Schema,添加Boolean类型    let bookSchema = new mongoose.Schema({      name:{        type:String,        required:true,        unique:true      },      author:{        type:String,        default:'匿名'      },      style:{        type:String,        enum:['言情','推理','小说','城市','战争']      },      price:Number,      is\_hot:Boolean,      tags:Array,      pub\_time:Date    })    //创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查    let bookModel = mongoose.model("books",bookSchema)    //新增数据 ，使用bookModel的create()方法    //查找一条数据，我们先在命令行把bookshop2的数据重新拷贝到bookshop，需要先删除bookshop数据库    // bookModel.findOne({name:"红楼梦"})    //     .then(data=>{    //       console.log(data);    //       mongoose.disconnect()    //     })    //     .catch(err=>{    //       console.log(err);    //       return    //   })    //根据id来查找    //  bookModel.findById('65dfbc6e10b2936a63de6204')    //     .then(data=>{    //       console.log(data);    //       mongoose.disconnect()    //     })    //     .catch(err=>{    //       console.log(err);    //       return    //     })    //查询多条数据，例如查找所有18.8元的书    //   bookModel.find({price:18.8})    //   .then(data=>{    //     console.log(data);    //     mongoose.disconnect()    //   })    //   .catch(err=>{    //     console.log(err);    //     return    // })    //查询所有文档数据，使用find()没有参数      bookModel.find().then(data=>{        console.log(data);        mongoose.disconnect()      }).catch(err=>{        console.log(err);        return      })  })  db.on('close',()=>{    console.log("关闭数据库...");  }) |

### 效果：能够查询到所有的数据

|  |
| --- |
| 数据库连接成功  [  {  \_id: new ObjectId('65dfafcb77c245bb92c6749c'),  name: '水浒传',  author: '施耐庵',  style: '城市',  price: 20.5,  is\_hot: true,  tags: [ '神话传说', '四大名著' ],  pub\_time: 2024-02-28T22:12:27.331Z,  \_\_v: 0  },  {  \_id: new ObjectId('65dfb023f37501f1bf2f5f6d'),  name: '三国演义',  author: '罗贯中',  style: '战争',  price: 19.8,  is\_hot: true,  tags: [ '神话传说', '四大名著' ],  pub\_time: 2024-02-28T22:13:55.453Z,  \_\_v: 0  },  {  \_id: new ObjectId('65dfbc6e10b2936a63de6201'),  name: '天龙八部',  author: '匿名',  style: '言情',  price: 18.8,  is\_hot: true,  tags: [ '神话传说', '武侠小说' ],  pub\_time: 2024-02-28T23:06:22.534Z,  \_\_v: 0  },  {  \_id: new ObjectId('65dfbc6e10b2936a63de6202'),  name: '射雕英雄传',  author: '匿名',  style: '言情',  price: 18.8,  is\_hot: true,  tags: [ '神话传说', '四大名著' ],  pub\_time: 2024-02-28T23:06:22.535Z,  \_\_v: 0  },  {  \_id: new ObjectId('65dfbc6e10b2936a63de6203'),  name: '金瓶梅',  author: '匿名',  style: '言情',  price: 18.8,  is\_hot: true,  tags: [ '神话传说', '四大禁书' ],  pub\_time: 2024-02-28T23:06:22.535Z,  \_\_v: 0  },  {  \_id: new ObjectId('65dfbc6e10b2936a63de6204'),  name: '倚天屠龙记',  author: '曹雪芹',  style: '言情',  price: 22.7,  is\_hot: true,  tags: [ '神话传说', '武侠小说' ],  pub\_time: 2024-02-28T23:06:22.535Z,  \_\_v: 0  },  {  \_id: new ObjectId('65dfdbdd122c223f0b4ca12c'),  name: '西游记',  author: '吴承恩',  price: 18.8,  is\_hot: true,  tags: [ '神话传说', '四大名著' ],  pub\_time: 2024-02-29T01:20:29.199Z,  \_\_v: 0  },  {  \_id: new ObjectId('65dfdcad2bfa98da0d55d82f'),  name: '红楼梦',  author: '曹雪芹',  style: '言情',  price: 18.8,  is\_hot: true,  tags: [ '神话传说', '四大名著' ],  pub\_time: 2024-02-29T01:23:57.603Z,  \_\_v: 0  }  ]  关闭数据库... |

# 9.mongoose条件控制，也就是条件表达式

## 我们上面学习的查找都是精准匹配，相当于用key=value的方式来查找，但是有些场景我们需要使用模糊查询，此时就需要使用条件，下面是mongoose常见的条件表达式

|  |
| --- |
|  |
|  |

## 案例，在进行条件查询之前，先运行00-insermany.js往数据库里面插入一些数据方便我们学习

|  |
| --- |
| //插入多条文档数据,promise语法  let mongoose = require('mongoose')  mongoose.connect('mongodb://127.0.0.1/bookshop')    //设置数据库回调事件处理  var db = mongoose.connection;  db.on('error', console.error.bind(console, 'connection error:')); //连接失败  db.once('open', function() { //连接成功    // we're connected!    console.log("数据库连接成功");    //所有的业务代码都需要在这里写，不能写在外面    //1.新增数据    //1.1先创建文档的结构对象Schema,添加Boolean类型    let bookSchema = new mongoose.Schema({      name:{        type:String,        required:true,        unique:true      },      author:{        type:String,        default:'匿名'      },      style:{        type:String,        enum:['言情','推理','小说','城市','战争','理财','思维']      },      price:Number,      is\_hot:Boolean,      tags:Array,      pub\_time:Date    })    //1.2创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查    let bookModel = mongoose.model("books",bookSchema)    //1.3新增多条数据 ，使用bookModel的insertMany()方法    let newBooks1 = [      {          name:"富爸爸21世纪的生意",          author:"罗伯特·清崎",          style:'理财',          price:17.5,          is\_hot:true,          tags:['理财','财商思维'],          pub\_time:new Date()     },      {          name:"决定孩子一生幸福的财商教育",          author:"卫裕峰 ",          style:'理财',          price:35.5,          is\_hot:true,          tags:['理财','财商思维'],          pub\_time:new Date()     },      {          name:"富爸爸财务自由之路",          author:"罗伯特·清崎",          style:'理财',          price:45,          is\_hot:true,          tags:['理财','财商思维'],          pub\_time:new Date()     },      {          name:"餐巾纸金融学：30秒内快速提高财商",          author:"蒂娜·海伊",          style:'理财',          price:30.8,          is\_hot:true,          tags:['理财','财商思维'],          pub\_time:new Date()     },      {          name:"个体突围：真正的高手，都有破局思维",          author:"艾玛・加侬",          style:'思维',          price:25,          is\_hot:true,          tags:['思维','心计'],          pub\_time:new Date()     },      {          name:"破局\_打造人才供应链",          author:"许锋",          style:'思维',          price:21,          is\_hot:true,          tags:['思维','心计'],          pub\_time:new Date()     },      {          name:"同桌的你",          author:"苏陌",          style:'言情',          price:25,          is\_hot:false,          tags:['言情','爱情小说'],          pub\_time:new Date()     },      {          name:"初恋",          author:"库列绍夫",          style:'言情',          price:10.8,          is\_hot:false,          tags:['言情','爱情小说'],          pub\_time:new Date()     },      {          name:"大诱拐",          author:"天藤真 ",          style:'推理',          price:9.8,          is\_hot:false,          tags:['推理','小说'],          pub\_time:new Date()     },      {          name:"月光森林",          author:"葵田谷",          style:'推理',          price:12.8,          is\_hot:false,          tags:['推理','小说'],          pub\_time:new Date()     },      {          name:"幸运数字3",          author:"玛格瑞.爱琳汉姆",          style:'推理',          price:22.8,          is\_hot:true,          tags:['推理','小说'],          pub\_time:new Date()     },      {          name:"谋杀的魅影",          author:"褚盟",          style:'推理',          price:25,          is\_hot:true,          tags:['推理','恐怖'],          pub\_time:new Date()     },    ]    // let newBooks = [    //   {    //       name:"天龙八部",    //       // author:"",    //       style:'言情',    //       price:18.8,    //       is\_hot:true,    //       tags:['神话传说','武侠小说'],    //       pub\_time:new Date()    //  },    //   {    //       name:"射雕英雄传",    //       // author:"曹雪芹",    //       style:'言情',    //       price:18.8,    //       is\_hot:true,    //       tags:['神话传说','四大名著'],    //       pub\_time:new Date()    //  },    //   {    //       name:"金瓶梅",    //       // author:"曹雪芹",    //       style:'言情',    //       price:18.8,    //       is\_hot:true,    //       tags:['神话传说','四大禁书'],    //       pub\_time:new Date()    //  },    //   {    //       name:"倚天屠龙记",    //       author:"曹雪芹",    //       style:'言情',    //       price:18.8,    //       is\_hot:true,    //       tags:['神话传说','武侠小说'],    //       pub\_time:new Date()    //  },    // ]    bookModel.insertMany(newBooks1).then(data=>{      console.log(data);      mongoose.disconnect()    })  })  db.on('close',()=>{    console.log("关闭数据库...");  }) |

### 运行成功，

|  |
| --- |
| 数据库连接成功  [  {  name: '富爸爸21世纪的生意',  author: '罗伯特·清崎',  style: '理财',  price: 17.5,  is\_hot: true,  tags: [ '理财', '财商思维' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_id: new ObjectId('65e0b8301a477c02feacda9d'),  \_\_v: 0  },  {  name: '决定孩子一生幸福的财商教育',  author: '卫裕峰 ',  style: '理财',  price: 35.5,  is\_hot: true,  tags: [ '理财', '财商思维' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_id: new ObjectId('65e0b8301a477c02feacda9e'),  \_\_v: 0  },  {  name: '富爸爸财务自由之路',  author: '罗伯特·清崎',  style: '理财',  price: 45,  is\_hot: true,  tags: [ '理财', '财商思维' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_id: new ObjectId('65e0b8301a477c02feacda9f'),  \_\_v: 0  },  {  name: '餐巾纸金融学：30秒内快速提高财商',  author: '蒂娜·海伊',  style: '理财',  price: 30.8,  is\_hot: true,  tags: [ '理财', '财商思维' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_id: new ObjectId('65e0b8301a477c02feacdaa0'),  \_\_v: 0  },  {  name: '个体突围：真正的高手，都有破局思维',  author: '艾玛・加侬',  style: '思维',  price: 25,  is\_hot: true,  tags: [ '思维', '心计' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_id: new ObjectId('65e0b8301a477c02feacdaa1'),  \_\_v: 0  },  {  name: '破局\_打造人才供应链',  author: '许锋',  style: '思维',  price: 21,  is\_hot: true,  tags: [ '思维', '心计' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_id: new ObjectId('65e0b8301a477c02feacdaa2'),  \_\_v: 0  },  {  name: '同桌的你',  author: '苏陌',  style: '言情',  price: 25,  is\_hot: false,  tags: [ '言情', '爱情小说' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_id: new ObjectId('65e0b8301a477c02feacdaa3'),  \_\_v: 0  },  {  name: '初恋',  author: '库列绍夫',  style: '言情',  price: 10.8,  is\_hot: false,  tags: [ '言情', '爱情小说' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_id: new ObjectId('65e0b8301a477c02feacdaa4'),  \_\_v: 0  },  {  name: '大诱拐',  author: '天藤真 ',  style: '推理',  price: 9.8,  is\_hot: false,  tags: [ '推理', '小说' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_id: new ObjectId('65e0b8301a477c02feacdaa5'),  \_\_v: 0  },  {  name: '月光森林',  author: '葵田谷',  style: '推理',  price: 12.8,  is\_hot: false,  tags: [ '推理', '小说' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_id: new ObjectId('65e0b8301a477c02feacdaa6'),  \_\_v: 0  },  {  name: '幸运数字3',  author: '玛格瑞.爱琳汉姆',  style: '推理',  price: 22.8,  is\_hot: true,  tags: [ '推理', '小说' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_id: new ObjectId('65e0b8301a477c02feacdaa7'),  \_\_v: 0  },  {  name: '谋杀的魅影',  author: '褚盟',  style: '推理',  price: 25,  is\_hot: true,  tags: [ '推理', '恐怖' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_id: new ObjectId('65e0b8301a477c02feacdaa8'),  \_\_v: 0  }  ]  关闭数据库... |

### 用命令行窗口查看，发现已经又20条数据了

|  |
| --- |
|  |

## 我们把这些数据复制一份到bookshop2

|  |
| --- |
|  |

## 我们新建一个08-mongoosbasic-conditional-search.js文件用来练习高级条件查询

## 1.比较运算符

### 1）$gt 表示大于

|  |
| --- |
| //查找数据，使用promise新语法  let mongoose = require('mongoose')  mongoose.connect('mongodb://127.0.0.1/bookshop')    //设置数据库回调事件处理  var db = mongoose.connection;  db.on('error', console.error.bind(console, 'connection error:')); //连接失败  db.once('open', function() { //连接成功    // we're connected!    console.log("数据库连接成功");    //所有的业务代码都需要在这里写，不能写在外面      //先创建文档的结构对象Schema,添加Boolean类型    let bookSchema = new mongoose.Schema({      name:{        type:String,        required:true,        unique:true      },      author:{        type:String,        default:'匿名'      },      style:{        type:String,        enum:['言情','推理','小说','城市','战争']      },      price:Number,      is\_hot:Boolean,      tags:Array,      pub\_time:Date    })    //创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查    let bookModel = mongoose.model("books",bookSchema)    //新增数据 ，使用bookModel的create()方法    //查找一条数据，我们先在命令行把bookshop2的数据重新拷贝到bookshop，需要先删除bookshop数据库    // bookModel.findOne({name:"红楼梦"})    //     .then(data=>{    //       console.log(data);    //       mongoose.disconnect()    //     })    //     .catch(err=>{    //       console.log(err);    //       return    //   })    //根据id来查找    //  bookModel.findById('65dfbc6e10b2936a63de6204')    //     .then(data=>{    //       console.log(data);    //       mongoose.disconnect()    //     })    //     .catch(err=>{    //       console.log(err);    //       return    //     })    //查询多条数据，例如查找所有18.8元的书    //   bookModel.find({price:18.8})    //   .then(data=>{    //     console.log(data);    //     mongoose.disconnect()    //   })    //   .catch(err=>{    //     console.log(err);    //     return    // })    //查询所有文档数据，使用find()没有参数      // bookModel.find().then(data=>{      //   console.log(data);      //   mongoose.disconnect()      // }).catch(err=>{      //   console.log(err);      //   return      // })        //条件查询，使用条件表达式      //1.$gt 表示大于      bookModel.find({"price":{$gt:15}})//查找价钱在15元以上的书           .then(data=>{            console.log(data);            mongoose.disconnect()           }).catch(err=>{            console.log(err);            mongoose.disconnect()            return        })  })  db.on('close',()=>{    console.log("关闭数据库...");  }) |

### 效果：

|  |
| --- |
| 数据库连接成功  [  {  \_id: new ObjectId('65dfafcb77c245bb92c6749c'),  name: '水浒传',  author: '施耐庵',  style: '城市',  price: 20.5,  is\_hot: true,  tags: [ '神话传说', '四大名著' ],  pub\_time: 2024-02-28T22:12:27.331Z,  \_\_v: 0  },  {  \_id: new ObjectId('65dfb023f37501f1bf2f5f6d'),  name: '三国演义',  author: '罗贯中',  style: '战争',  price: 19.8,  is\_hot: true,  tags: [ '神话传说', '四大名著' ],  pub\_time: 2024-02-28T22:13:55.453Z,  \_\_v: 0  },  {  \_id: new ObjectId('65dfbc6e10b2936a63de6201'),  name: '天龙八部',  author: '匿名',  style: '言情',  price: 18.8,  is\_hot: true,  tags: [ '神话传说', '武侠小说' ],  pub\_time: 2024-02-28T23:06:22.534Z,  \_\_v: 0  },  {  \_id: new ObjectId('65dfbc6e10b2936a63de6202'),  name: '射雕英雄传',  author: '匿名',  style: '言情',  price: 18.8,  is\_hot: true,  tags: [ '神话传说', '四大名著' ],  pub\_time: 2024-02-28T23:06:22.535Z,  \_\_v: 0  },  {  \_id: new ObjectId('65dfbc6e10b2936a63de6203'),  name: '金瓶梅',  author: '匿名',  style: '言情',  price: 18.8,  is\_hot: true,  tags: [ '神话传说', '四大禁书' ],  pub\_time: 2024-02-28T23:06:22.535Z,  \_\_v: 0  },  {  \_id: new ObjectId('65dfbc6e10b2936a63de6204'),  name: '倚天屠龙记',  author: '曹雪芹',  style: '言情',  price: 22.7,  is\_hot: true,  tags: [ '神话传说', '武侠小说' ],  pub\_time: 2024-02-28T23:06:22.535Z,  \_\_v: 0  },  {  \_id: new ObjectId('65dfdbdd122c223f0b4ca12c'),  name: '西游记',  author: '吴承恩',  price: 18.8,  is\_hot: true,  tags: [ '神话传说', '四大名著' ],  pub\_time: 2024-02-29T01:20:29.199Z,  \_\_v: 0  },  {  \_id: new ObjectId('65dfdcad2bfa98da0d55d82f'),  name: '红楼梦',  author: '曹雪芹',  style: '言情',  price: 18.8,  is\_hot: true,  tags: [ '神话传说', '四大名著' ],  pub\_time: 2024-02-29T01:23:57.603Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacda9d'),  name: '富爸爸21世纪的生意',  author: '罗伯特·清崎',  style: '理财',  price: 17.5,  is\_hot: true,  tags: [ '理财', '财商思维' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacda9e'),  name: '决定孩子一生幸福的财商教育',  author: '卫裕峰 ',  style: '理财',  price: 35.5,  is\_hot: true,  tags: [ '理财', '财商思维' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacda9f'),  name: '富爸爸财务自由之路',  author: '罗伯特·清崎',  style: '理财',  price: 45,  is\_hot: true,  tags: [ '理财', '财商思维' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa0'),  name: '餐巾纸金融学：30秒内快速提高财商',  author: '蒂娜·海伊',  style: '理财',  price: 30.8,  is\_hot: true,  tags: [ '理财', '财商思维' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa1'),  name: '个体突围：真正的高手，都有破局思维',  author: '艾玛・加侬',  style: '思维',  price: 25,  is\_hot: true,  tags: [ '思维', '心计' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa2'),  name: '破局\_打造人才供应链',  author: '许锋',  style: '思维',  price: 21,  is\_hot: true,  tags: [ '思维', '心计' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa3'),  name: '同桌的你',  author: '苏陌',  style: '言情',  price: 25,  is\_hot: false,  tags: [ '言情', '爱情小说' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa7'),  name: '幸运数字3',  author: '玛格瑞.爱琳汉姆',  style: '推理',  price: 22.8,  is\_hot: true,  tags: [ '推理', '小说' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa8'),  name: '谋杀的魅影',  author: '褚盟',  style: '推理',  price: 25,  is\_hot: true,  tags: [ '推理', '恐怖' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  }  ]  关闭数据库... |

### 2）$gte　　　 大于等于

|  |
| --- |
| //查找数据，使用promise新语法  let mongoose = require('mongoose')  mongoose.connect('mongodb://127.0.0.1/bookshop')    //设置数据库回调事件处理  var db = mongoose.connection;  db.on('error', console.error.bind(console, 'connection error:')); //连接失败  db.once('open', function() { //连接成功    // we're connected!    console.log("数据库连接成功");    //所有的业务代码都需要在这里写，不能写在外面      //先创建文档的结构对象Schema,添加Boolean类型    let bookSchema = new mongoose.Schema({      name:{        type:String,        required:true,        unique:true      },      author:{        type:String,        default:'匿名'      },      style:{        type:String,        enum:['言情','推理','小说','城市','战争']      },      price:Number,      is\_hot:Boolean,      tags:Array,      pub\_time:Date    })    //创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查    let bookModel = mongoose.model("books",bookSchema)    //新增数据 ，使用bookModel的create()方法    //查找一条数据，我们先在命令行把bookshop2的数据重新拷贝到bookshop，需要先删除bookshop数据库    // bookModel.findOne({name:"红楼梦"})    //     .then(data=>{    //       console.log(data);    //       mongoose.disconnect()    //     })    //     .catch(err=>{    //       console.log(err);    //       return    //   })    //根据id来查找    //  bookModel.findById('65dfbc6e10b2936a63de6204')    //     .then(data=>{    //       console.log(data);    //       mongoose.disconnect()    //     })    //     .catch(err=>{    //       console.log(err);    //       return    //     })    //查询多条数据，例如查找所有18.8元的书    //   bookModel.find({price:18.8})    //   .then(data=>{    //     console.log(data);    //     mongoose.disconnect()    //   })    //   .catch(err=>{    //     console.log(err);    //     return    // })    //查询所有文档数据，使用find()没有参数      // bookModel.find().then(data=>{      //   console.log(data);      //   mongoose.disconnect()      // }).catch(err=>{      //   console.log(err);      //   return      // })        //高级条件查询，使用条件表达式      //2.$gte 表示大于等于      bookModel.find({"price":{$gte:25}})//查找价钱在25元或者以上的书           .then(data=>{            console.log(data);            mongoose.disconnect()           }).catch(err=>{            console.log(err);            mongoose.disconnect()            return        })  })  db.on('close',()=>{    console.log("关闭数据库...");  }) |

### 效果：

|  |
| --- |
| 数据库连接成功  [  {  \_id: new ObjectId('65e0b8301a477c02feacda9e'),  name: '决定孩子一生幸福的财商教育',  author: '卫裕峰 ',  style: '理财',  price: 35.5,  is\_hot: true,  tags: [ '理财', '财商思维' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacda9f'),  name: '富爸爸财务自由之路',  author: '罗伯特·清崎',  style: '理财',  price: 45,  is\_hot: true,  tags: [ '理财', '财商思维' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa0'),  name: '餐巾纸金融学：30秒内快速提高财商',  author: '蒂娜·海伊',  style: '理财',  price: 30.8,  is\_hot: true,  tags: [ '理财', '财商思维' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa1'),  name: '个体突围：真正的高手，都有破局思维',  author: '艾玛・加侬',  style: '思维',  price: 25,  is\_hot: true,  tags: [ '思维', '心计' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa3'),  name: '同桌的你',  author: '苏陌',  style: '言情',  price: 25,  is\_hot: false,  tags: [ '言情', '爱情小说' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa8'),  name: '谋杀的魅影',  author: '褚盟',  style: '推理',  price: 25,  is\_hot: true,  tags: [ '推理', '恐怖' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  }  ]  关闭数据库... |

### 3）　$lt　　　　 小于 $lte　　　 小于等于和上面类似，不演示了

### 4）组合写法例如查找所有价钱在15到25元之间的书

|  |
| --- |
| //查找数据，使用promise新语法  let mongoose = require('mongoose')  mongoose.connect('mongodb://127.0.0.1/bookshop')    //设置数据库回调事件处理  var db = mongoose.connection;  db.on('error', console.error.bind(console, 'connection error:')); //连接失败  db.once('open', function() { //连接成功    // we're connected!    console.log("数据库连接成功");    //所有的业务代码都需要在这里写，不能写在外面      //先创建文档的结构对象Schema,添加Boolean类型    let bookSchema = new mongoose.Schema({      name:{        type:String,        required:true,        unique:true      },      author:{        type:String,        default:'匿名'      },      style:{        type:String,        enum:['言情','推理','小说','城市','战争']      },      price:Number,      is\_hot:Boolean,      tags:Array,      pub\_time:Date    })    //创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查    let bookModel = mongoose.model("books",bookSchema)    //新增数据 ，使用bookModel的create()方法    //查找一条数据，我们先在命令行把bookshop2的数据重新拷贝到bookshop，需要先删除bookshop数据库    // bookModel.findOne({name:"红楼梦"})    //     .then(data=>{    //       console.log(data);    //       mongoose.disconnect()    //     })    //     .catch(err=>{    //       console.log(err);    //       return    //   })    //根据id来查找    //  bookModel.findById('65dfbc6e10b2936a63de6204')    //     .then(data=>{    //       console.log(data);    //       mongoose.disconnect()    //     })    //     .catch(err=>{    //       console.log(err);    //       return    //     })    //查询多条数据，例如查找所有18.8元的书    //   bookModel.find({price:18.8})    //   .then(data=>{    //     console.log(data);    //     mongoose.disconnect()    //   })    //   .catch(err=>{    //     console.log(err);    //     return    // })    //查询所有文档数据，使用find()没有参数      // bookModel.find().then(data=>{      //   console.log(data);      //   mongoose.disconnect()      // }).catch(err=>{      //   console.log(err);      //   return      // })        //高级条件查询，使用条件表达式      //组合写法例如查找所有价钱在15到25元之间的书      bookModel.find({"price":{$gte:15,$lte:25}})           .then(data=>{            console.log(data);            mongoose.disconnect()           }).catch(err=>{            console.log(err);            mongoose.disconnect()            return        })  })  db.on('close',()=>{    console.log("关闭数据库...");  }) |

### 还可以这么写

|  |
| --- |
| //查找数据，使用promise新语法  let mongoose = require('mongoose')  mongoose.connect('mongodb://127.0.0.1/bookshop')    //设置数据库回调事件处理  var db = mongoose.connection;  db.on('error', console.error.bind(console, 'connection error:')); //连接失败  db.once('open', function() { //连接成功    // we're connected!    console.log("数据库连接成功");    //所有的业务代码都需要在这里写，不能写在外面      //先创建文档的结构对象Schema,添加Boolean类型    let bookSchema = new mongoose.Schema({      name:{        type:String,        required:true,        unique:true      },      author:{        type:String,        default:'匿名'      },      style:{        type:String,        enum:['言情','推理','小说','城市','战争']      },      price:Number,      is\_hot:Boolean,      tags:Array,      pub\_time:Date    })    //创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查    let bookModel = mongoose.model("books",bookSchema)    //新增数据 ，使用bookModel的create()方法    //查找一条数据，我们先在命令行把bookshop2的数据重新拷贝到bookshop，需要先删除bookshop数据库    // bookModel.findOne({name:"红楼梦"})    //     .then(data=>{    //       console.log(data);    //       mongoose.disconnect()    //     })    //     .catch(err=>{    //       console.log(err);    //       return    //   })    //根据id来查找    //  bookModel.findById('65dfbc6e10b2936a63de6204')    //     .then(data=>{    //       console.log(data);    //       mongoose.disconnect()    //     })    //     .catch(err=>{    //       console.log(err);    //       return    //     })    //查询多条数据，例如查找所有18.8元的书    //   bookModel.find({price:18.8})    //   .then(data=>{    //     console.log(data);    //     mongoose.disconnect()    //   })    //   .catch(err=>{    //     console.log(err);    //     return    // })    //查询所有文档数据，使用find()没有参数      // bookModel.find().then(data=>{      //   console.log(data);      //   mongoose.disconnect()      // }).catch(err=>{      //   console.log(err);      //   return      // })        //高级条件查询，使用条件表达式      //组合写法例如查找所有价钱在15到25元之间的书      // bookModel.find({"price":{$gte:15,$lte:25}})      //      .then(data=>{      //       console.log(data);      //       mongoose.disconnect()      //      }).catch(err=>{      //       console.log(err);      //       mongoose.disconnect()      //       return      //   })      bookModel.where({"price":{$gte:15,$lte:25}})           .then(data=>{            console.log(data);            mongoose.disconnect()           }).catch(err=>{            console.log(err);            mongoose.disconnect()            return        })  })  db.on('close',()=>{    console.log("关闭数据库...");  }) |

### 或者这么写

|  |
| --- |
| //查找数据，使用promise新语法  let mongoose = require('mongoose')  mongoose.connect('mongodb://127.0.0.1/bookshop')    //设置数据库回调事件处理  var db = mongoose.connection;  db.on('error', console.error.bind(console, 'connection error:')); //连接失败  db.once('open', function() { //连接成功    // we're connected!    console.log("数据库连接成功");    //所有的业务代码都需要在这里写，不能写在外面      //先创建文档的结构对象Schema,添加Boolean类型    let bookSchema = new mongoose.Schema({      name:{        type:String,        required:true,        unique:true      },      author:{        type:String,        default:'匿名'      },      style:{        type:String,        enum:['言情','推理','小说','城市','战争']      },      price:Number,      is\_hot:Boolean,      tags:Array,      pub\_time:Date    })    //创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查    let bookModel = mongoose.model("books",bookSchema)    //新增数据 ，使用bookModel的create()方法    //查找一条数据，我们先在命令行把bookshop2的数据重新拷贝到bookshop，需要先删除bookshop数据库    // bookModel.findOne({name:"红楼梦"})    //     .then(data=>{    //       console.log(data);    //       mongoose.disconnect()    //     })    //     .catch(err=>{    //       console.log(err);    //       return    //   })    //根据id来查找    //  bookModel.findById('65dfbc6e10b2936a63de6204')    //     .then(data=>{    //       console.log(data);    //       mongoose.disconnect()    //     })    //     .catch(err=>{    //       console.log(err);    //       return    //     })    //查询多条数据，例如查找所有18.8元的书    //   bookModel.find({price:18.8})    //   .then(data=>{    //     console.log(data);    //     mongoose.disconnect()    //   })    //   .catch(err=>{    //     console.log(err);    //     return    // })    //查询所有文档数据，使用find()没有参数      // bookModel.find().then(data=>{      //   console.log(data);      //   mongoose.disconnect()      // }).catch(err=>{      //   console.log(err);      //   return      // })        //高级条件查询，使用条件表达式  //组合写法例如查找所有价钱在15到25元之间的书  //写法1      // bookModel.find({"price":{$gte:15,$lte:25}})      //      .then(data=>{      //       console.log(data);      //       mongoose.disconnect()      //      }).catch(err=>{      //       console.log(err);      //       mongoose.disconnect()      //       return  //   })  //写法2      // bookModel.where({"price":{$gte:15,$lte:25}})      //      .then(data=>{      //       console.log(data);      //       mongoose.disconnect()      //      }).catch(err=>{      //       console.log(err);      //       mongoose.disconnect()      //       return      //   })  //写法3      bookModel.where("price").gte(15).lte(25)           .then(data=>{            console.log(data);            mongoose.disconnect()           }).catch(err=>{            console.log(err);            mongoose.disconnect()            return        })  })  db.on('close',()=>{    console.log("关闭数据库...");  }) |

### 效果：

|  |
| --- |
| 数据库连接成功  [  {  \_id: new ObjectId('65dfafcb77c245bb92c6749c'),  name: '水浒传',  author: '施耐庵',  style: '城市',  price: 20.5,  is\_hot: true,  tags: [ '神话传说', '四大名著' ],  pub\_time: 2024-02-28T22:12:27.331Z,  \_\_v: 0  },  {  \_id: new ObjectId('65dfb023f37501f1bf2f5f6d'),  name: '三国演义',  author: '罗贯中',  style: '战争',  price: 19.8,  is\_hot: true,  tags: [ '神话传说', '四大名著' ],  pub\_time: 2024-02-28T22:13:55.453Z,  \_\_v: 0  },  {  \_id: new ObjectId('65dfbc6e10b2936a63de6201'),  name: '天龙八部',  author: '匿名',  style: '言情',  price: 18.8,  is\_hot: true,  tags: [ '神话传说', '武侠小说' ],  pub\_time: 2024-02-28T23:06:22.534Z,  \_\_v: 0  },  {  \_id: new ObjectId('65dfbc6e10b2936a63de6202'),  name: '射雕英雄传',  author: '匿名',  style: '言情',  price: 18.8,  is\_hot: true,  tags: [ '神话传说', '四大名著' ],  pub\_time: 2024-02-28T23:06:22.535Z,  \_\_v: 0  },  {  \_id: new ObjectId('65dfbc6e10b2936a63de6203'),  name: '金瓶梅',  author: '匿名',  style: '言情',  price: 18.8,  is\_hot: true,  tags: [ '神话传说', '四大禁书' ],  pub\_time: 2024-02-28T23:06:22.535Z,  \_\_v: 0  },  {  \_id: new ObjectId('65dfbc6e10b2936a63de6204'),  name: '倚天屠龙记',  author: '曹雪芹',  style: '言情',  price: 22.7,  is\_hot: true,  tags: [ '神话传说', '武侠小说' ],  pub\_time: 2024-02-28T23:06:22.535Z,  \_\_v: 0  },  {  \_id: new ObjectId('65dfdbdd122c223f0b4ca12c'),  name: '西游记',  author: '吴承恩',  price: 18.8,  is\_hot: true,  tags: [ '神话传说', '四大名著' ],  pub\_time: 2024-02-29T01:20:29.199Z,  \_\_v: 0  },  {  \_id: new ObjectId('65dfdcad2bfa98da0d55d82f'),  name: '红楼梦',  author: '曹雪芹',  style: '言情',  price: 18.8,  is\_hot: true,  tags: [ '神话传说', '四大名著' ],  pub\_time: 2024-02-29T01:23:57.603Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacda9d'),  name: '富爸爸21世纪的生意',  author: '罗伯特·清崎',  style: '理财',  price: 17.5,  is\_hot: true,  tags: [ '理财', '财商思维' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa1'),  name: '个体突围：真正的高手，都有破局思维',  author: '艾玛・加侬',  style: '思维',  price: 25,  is\_hot: true,  tags: [ '思维', '心计' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa2'),  name: '破局\_打造人才供应链',  author: '许锋',  style: '思维',  price: 21,  is\_hot: true,  tags: [ '思维', '心计' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa3'),  name: '同桌的你',  author: '苏陌',  style: '言情',  price: 25,  is\_hot: false,  tags: [ '言情', '爱情小说' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa7'),  name: '幸运数字3',  author: '玛格瑞.爱琳汉姆',  style: '推理',  price: 22.8,  is\_hot: true,  tags: [ '推理', '小说' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa8'),  name: '谋杀的魅影',  author: '褚盟',  style: '推理',  price: 25,  is\_hot: true,  tags: [ '推理', '恐怖' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  }  ]  关闭数据库... |

### 5）$ne,例如查找所有is\_hot不是true的书

|  |
| --- |
| //查找数据，使用promise新语法  let mongoose = require('mongoose')  mongoose.connect('mongodb://127.0.0.1/bookshop')    //设置数据库回调事件处理  var db = mongoose.connection;  db.on('error', console.error.bind(console, 'connection error:')); //连接失败  db.once('open', function() { //连接成功    // we're connected!    console.log("数据库连接成功");    //所有的业务代码都需要在这里写，不能写在外面      //先创建文档的结构对象Schema,添加Boolean类型    let bookSchema = new mongoose.Schema({      name:{        type:String,        required:true,        unique:true      },      author:{        type:String,        default:'匿名'      },      style:{        type:String,        enum:['言情','推理','小说','城市','战争']      },      price:Number,      is\_hot:Boolean,      tags:Array,      pub\_time:Date    })    //创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查    let bookModel = mongoose.model("books",bookSchema)    //新增数据 ，使用bookModel的create()方法    //查找一条数据，我们先在命令行把bookshop2的数据重新拷贝到bookshop，需要先删除bookshop数据库    // bookModel.findOne({name:"红楼梦"})    //     .then(data=>{    //       console.log(data);    //       mongoose.disconnect()    //     })    //     .catch(err=>{    //       console.log(err);    //       return    //   })    //根据id来查找    //  bookModel.findById('65dfbc6e10b2936a63de6204')    //     .then(data=>{    //       console.log(data);    //       mongoose.disconnect()    //     })    //     .catch(err=>{    //       console.log(err);    //       return    //     })    //查询多条数据，例如查找所有18.8元的书    //   bookModel.find({price:18.8})    //   .then(data=>{    //     console.log(data);    //     mongoose.disconnect()    //   })    //   .catch(err=>{    //     console.log(err);    //     return    // })    //查询所有文档数据，使用find()没有参数      // bookModel.find().then(data=>{      //   console.log(data);      //   mongoose.disconnect()      // }).catch(err=>{      //   console.log(err);      //   return      // })        //高级条件查询，使用条件表达式      //$ne,例如查找所有is\_hot不是true的书      bookModel.find({"is\_hot":{$ne:true}})        .then(data=>{          console.log(data);          mongoose.disconnect()        }).catch(err=>{          console.log(err);          mongoose.disconnect()          return        })    })  db.on('close',()=>{    console.log("关闭数据库...");  }) |

### 也可以这么写

|  |
| --- |
| //查找数据，使用promise新语法  let mongoose = require('mongoose')  mongoose.connect('mongodb://127.0.0.1/bookshop')    //设置数据库回调事件处理  var db = mongoose.connection;  db.on('error', console.error.bind(console, 'connection error:')); //连接失败  db.once('open', function() { //连接成功    // we're connected!    console.log("数据库连接成功");    //所有的业务代码都需要在这里写，不能写在外面      //先创建文档的结构对象Schema,添加Boolean类型    let bookSchema = new mongoose.Schema({      name:{        type:String,        required:true,        unique:true      },      author:{        type:String,        default:'匿名'      },      style:{        type:String,        enum:['言情','推理','小说','城市','战争']      },      price:Number,      is\_hot:Boolean,      tags:Array,      pub\_time:Date    })    //创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查    let bookModel = mongoose.model("books",bookSchema)    //新增数据 ，使用bookModel的create()方法    //查找一条数据，我们先在命令行把bookshop2的数据重新拷贝到bookshop，需要先删除bookshop数据库  //高级条件查询，使用条件表达式        bookModel.where("is\_hot").ne(true)        .then(data=>{          console.log(data);          mongoose.disconnect()        }).catch(err=>{          console.log(err);          mongoose.disconnect()          return        })    })  db.on('close',()=>{    console.log("关闭数据库...");  }) |

### 效果

|  |
| --- |
| 数据库连接成功  [  {  \_id: new ObjectId('65e0b8301a477c02feacdaa3'),  name: '同桌的你',  author: '苏陌',  style: '言情',  price: 25,  is\_hot: false,  tags: [ '言情', '爱情小说' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa4'),  name: '初恋',  author: '库列绍夫',  style: '言情',  price: 10.8,  is\_hot: false,  tags: [ '言情', '爱情小说' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa5'),  name: '大诱拐',  author: '天藤真 ',  style: '推理',  price: 9.8,  is\_hot: false,  tags: [ '推理', '小说' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa6'),  name: '月光森林',  author: '葵田谷',  style: '推理',  price: 12.8,  is\_hot: false,  tags: [ '推理', '小说' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  }  ]  关闭数据库... |

### 6）查询条件为数组，也叫做数组

|  |
| --- |
| //查找数据，使用promise新语法  let mongoose = require('mongoose')  mongoose.connect('mongodb://127.0.0.1/bookshop')    //设置数据库回调事件处理  var db = mongoose.connection;  db.on('error', console.error.bind(console, 'connection error:')); //连接失败  db.once('open', function() { //连接成功    // we're connected!    console.log("数据库连接成功");    //所有的业务代码都需要在这里写，不能写在外面      //先创建文档的结构对象Schema,添加Boolean类型    let bookSchema = new mongoose.Schema({      name:{        type:String,        required:true,        unique:true      },      author:{        type:String,        default:'匿名'      },      style:{        type:String,        enum:['言情','推理','小说','城市','战争']      },      price:Number,      is\_hot:Boolean,      tags:Array,      pub\_time:Date    })    //创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查    let bookModel = mongoose.model("books",bookSchema)    //新增数据 ，使用bookModel的create()方法    //查询条件为数组    bookModel.find({"price":[15,25]}) //只要符合其中一个就可以了           .then(data=>{            console.log(data);            mongoose.disconnect()           }).catch(err=>{            console.log(err);            mongoose.disconnect()            return        })     //$or高级查询，如查询价钱为15或者25的书  })  db.on('close',()=>{    console.log("关闭数据库...");  }) |

### 效果：

|  |
| --- |
| 数据库连接成功  [  {  \_id: new ObjectId('65e0b8301a477c02feacdaa1'),  name: '个体突围：真正的高手，都有破局思维',  author: '艾玛・加侬',  style: '思维',  price: 25,  is\_hot: true,  tags: [ '思维', '心计' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa3'),  name: '同桌的你',  author: '苏陌',  style: '言情',  price: 25,  is\_hot: false,  tags: [ '言情', '爱情小说' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa8'),  name: '谋杀的魅影',  author: '褚盟',  style: '推理',  price: 25,  is\_hot: true,  tags: [ '推理', '恐怖' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  }  ]  关闭数据库... |

## 2.逻辑运算符

### 1）$or

#### a.查询所有价格等于15或者价格等于25的书

|  |
| --- |
| //查找数据，使用promise新语法  let mongoose = require('mongoose')  mongoose.connect('mongodb://127.0.0.1/bookshop')    //设置数据库回调事件处理  var db = mongoose.connection;  db.on('error', console.error.bind(console, 'connection error:')); //连接失败  db.once('open', function() { //连接成功    // we're connected!    console.log("数据库连接成功");    //所有的业务代码都需要在这里写，不能写在外面      //先创建文档的结构对象Schema,添加Boolean类型    let bookSchema = new mongoose.Schema({      name:{        type:String,        required:true,        unique:true      },      author:{        type:String,        default:'匿名'      },      style:{        type:String,        enum:['言情','推理','小说','城市','战争']      },      price:Number,      is\_hot:Boolean,      tags:Array,      pub\_time:Date    })    //创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查    let bookModel = mongoose.model("books",bookSchema)    //新增数据 ，使用bookModel的create()方法       //$or高级查询，如查询价钱为15或者25的书     bookModel.find({$or:[{price:15},{price:25}]})         .then(data=>{          console.log(data);          mongoose.disconnect()         }).catch(err=>{          console.log(err);          mongoose.disconnect() //实际开发中不用经常disconnect，这里是方便调试而已          return        })  })  db.on('close',()=>{    console.log("关闭数据库...");  }) |

### 效果：

|  |
| --- |
| 数据库连接成功  [  {  \_id: new ObjectId('65e0b8301a477c02feacdaa1'),  name: '个体突围：真正的高手，都有破局思维',  author: '艾玛・加侬',  style: '思维',  price: 25,  is\_hot: true,  tags: [ '思维', '心计' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa3'),  name: '同桌的你',  author: '苏陌',  style: '言情',  price: 25,  is\_hot: false,  tags: [ '言情', '爱情小说' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa8'),  name: '谋杀的魅影',  author: '褚盟',  style: '推理',  price: 25,  is\_hot: true,  tags: [ '推理', '恐怖' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  }  ]  关闭数据库... |

#### b. 如查询价钱小于15或者大于25的书

|  |
| --- |
| //查找数据，使用promise新语法  let mongoose = require('mongoose')  mongoose.connect('mongodb://127.0.0.1/bookshop')    //设置数据库回调事件处理  var db = mongoose.connection;  db.on('error', console.error.bind(console, 'connection error:')); //连接失败  db.once('open', function() { //连接成功    // we're connected!    console.log("数据库连接成功");    //所有的业务代码都需要在这里写，不能写在外面      //先创建文档的结构对象Schema,添加Boolean类型    let bookSchema = new mongoose.Schema({      name:{        type:String,        required:true,        unique:true      },      author:{        type:String,        default:'匿名'      },      style:{        type:String,        enum:['言情','推理','小说','城市','战争']      },      price:Number,      is\_hot:Boolean,      tags:Array,      pub\_time:Date    })    //创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查    let bookModel = mongoose.model("books",bookSchema)    //新增数据 ，使用bookModel的create()方法       //$or高级查询，如查询价钱为15或者25的书    //  bookModel.find({$or:[{price:15},{price:25}]})    //      .then(data=>{    //       console.log(data);    //       mongoose.disconnect()    //      }).catch(err=>{    //       console.log(err);    //       mongoose.disconnect() //实际开发中不用经常disconnect，这里是方便调试而已    //       return    //     })    // 如查询价钱小于15或者大于25的书     bookModel.find({$or:[{price:{$lt:15}},{price:{$gt:25}}]})          .then(data=>{            console.log(data);            mongoose.disconnect()          }).catch(err=>{            console.log(err);            mongoose.disconnect()            return        })  })  db.on('close',()=>{    console.log("关闭数据库...");  }) |

### 效果：

|  |
| --- |
| 数据库连接成功  [  {  \_id: new ObjectId('65e0b8301a477c02feacda9e'),  name: '决定孩子一生幸福的财商教育',  author: '卫裕峰 ',  style: '理财',  price: 35.5,  is\_hot: true,  tags: [ '理财', '财商思维' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacda9f'),  name: '富爸爸财务自由之路',  author: '罗伯特·清崎',  style: '理财',  price: 45,  is\_hot: true,  tags: [ '理财', '财商思维' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa0'),  name: '餐巾纸金融学：30秒内快速提高财商',  author: '蒂娜·海伊',  style: '理财',  price: 30.8,  is\_hot: true,  tags: [ '理财', '财商思维' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa4'),  name: '初恋',  author: '库列绍夫',  style: '言情',  price: 10.8,  is\_hot: false,  tags: [ '言情', '爱情小说' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa5'),  name: '大诱拐',  author: '天藤真 ',  style: '推理',  price: 9.8,  is\_hot: false,  tags: [ '推理', '小说' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa6'),  name: '月光森林',  author: '葵田谷',  style: '推理',  price: 12.8,  is\_hot: false,  tags: [ '推理', '小说' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  }  ]  关闭数据库... |

### 2）$and如：查找is\_hot为false并且价格小于22的书

|  |
| --- |
| //查找数据，使用promise新语法  let mongoose = require('mongoose')  mongoose.connect('mongodb://127.0.0.1/bookshop')    //设置数据库回调事件处理  var db = mongoose.connection;  db.on('error', console.error.bind(console, 'connection error:')); //连接失败  db.once('open', function() { //连接成功    // we're connected!    console.log("数据库连接成功");    //所有的业务代码都需要在这里写，不能写在外面      //先创建文档的结构对象Schema,添加Boolean类型    let bookSchema = new mongoose.Schema({      name:{        type:String,        required:true,        unique:true      },      author:{        type:String,        default:'匿名'      },      style:{        type:String,        enum:['言情','推理','小说','城市','战争']      },      price:Number,      is\_hot:Boolean,      tags:Array,      pub\_time:Date    })    //创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查    let bookModel = mongoose.model("books",bookSchema)    //新增数据 ，使用bookModel的create()方法    //$and 查找is\_hot为false并且价格小于22的书    bookModel.find({$and:[{is\_hot:false},{price:{$lt:22}}]})        .then(data=>{          console.log(data);          mongoose.disconnect()        }).catch(err=>{          console.log(err);          mongoose.disconnect()          return      })  })  db.on('close',()=>{    console.log("关闭数据库...");  }) |

### 效果

|  |
| --- |
| 数据库连接成功  [  {  \_id: new ObjectId('65e0b8301a477c02feacdaa4'),  name: '初恋',  author: '库列绍夫',  style: '言情',  price: 10.8,  is\_hot: false,  tags: [ '言情', '爱情小说' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa5'),  name: '大诱拐',  author: '天藤真 ',  style: '推理',  price: 9.8,  is\_hot: false,  tags: [ '推理', '小说' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa6'),  name: '月光森林',  author: '葵田谷',  style: '推理',  price: 12.8,  is\_hot: false,  tags: [ '推理', '小说' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  }  ]  关闭数据库... |

## 3. 正则表达式条件查询，如查询书名包含'富'的书

|  |
| --- |
| //查找数据，使用promise新语法  let mongoose = require('mongoose')  mongoose.connect('mongodb://127.0.0.1/bookshop')    //设置数据库回调事件处理  var db = mongoose.connection;  db.on('error', console.error.bind(console, 'connection error:')); //连接失败  db.once('open', function() { //连接成功    // we're connected!    console.log("数据库连接成功");    //所有的业务代码都需要在这里写，不能写在外面      //先创建文档的结构对象Schema,添加Boolean类型    let bookSchema = new mongoose.Schema({      name:{        type:String,        required:true,        unique:true      },      author:{        type:String,        default:'匿名'      },      style:{        type:String,        enum:['言情','推理','小说','城市','战争']      },      price:Number,      is\_hot:Boolean,      tags:Array,      pub\_time:Date    })    //创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查    let bookModel = mongoose.model("books",bookSchema)    //新增数据 ，使用bookModel的create()方法    //正则表达式条件查询，如查询书名包含'富'的书    bookModel.find({name:/富/})    .then(data=>{      console.log(data);      mongoose.disconnect()    }).catch(err=>{      console.log(err);      mongoose.disconnect()      return  })  })  db.on('close',()=>{    console.log("关闭数据库...");  }) |

### 也可以这么写

|  |
| --- |
| //查找数据，使用promise新语法  let mongoose = require('mongoose')  mongoose.connect('mongodb://127.0.0.1/bookshop')    //设置数据库回调事件处理  var db = mongoose.connection;  db.on('error', console.error.bind(console, 'connection error:')); //连接失败  db.once('open', function() { //连接成功    // we're connected!    console.log("数据库连接成功");    //所有的业务代码都需要在这里写，不能写在外面      //先创建文档的结构对象Schema,添加Boolean类型    let bookSchema = new mongoose.Schema({      name:{        type:String,        required:true,        unique:true      },      author:{        type:String,        default:'匿名'      },      style:{        type:String,        enum:['言情','推理','小说','城市','战争']      },      price:Number,      is\_hot:Boolean,      tags:Array,      pub\_time:Date    })    //创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查    let bookModel = mongoose.model("books",bookSchema)    //新增数据 ，使用bookModel的create()方法    //正则表达式条件查询，如查询书名包含'富'的书    //   bookModel.find({name:/富/})    //   .then(data=>{    //     console.log(data);    //     mongoose.disconnect()    //   }).catch(err=>{    //     console.log(err);    //     mongoose.disconnect()    //     return    // })      bookModel.find({name:new RegExp('富')})      .then(data=>{        console.log(data);        mongoose.disconnect()      }).catch(err=>{        console.log(err);        mongoose.disconnect()        return    })  })  db.on('close',()=>{    console.log("关闭数据库...");  }) |

### 效果

|  |
| --- |
| 数据库连接成功  [  {  \_id: new ObjectId('65e0b8301a477c02feacda9d'),  name: '富爸爸21世纪的生意',  author: '罗伯特·清崎',  style: '理财',  price: 17.5,  is\_hot: true,  tags: [ '理财', '财商思维' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacda9f'),  name: '富爸爸财务自由之路',  author: '罗伯特·清崎',  style: '理财',  price: 45,  is\_hot: true,  tags: [ '理财', '财商思维' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  }  ]  ]  关闭数据库... |

# 10.个性化查询

## 1）select用来选择字段，0表示不要的字段1表示要的字段

|  |
| --- |
| //mongoose查询后处理，promise语法  let mongoose = require('mongoose')  mongoose.connect('mongodb://127.0.0.1/bookshop')    //设置数据库回调事件处理  var db = mongoose.connection;  db.on('error', console.error.bind(console, 'connection error:')); //连接失败  db.once('open', function() { //连接成功    // we're connected!    console.log("数据库连接成功");    //所有的业务代码都需要在这里写，不能写在外面      //先创建文档的结构对象Schema,添加Boolean类型    let bookSchema = new mongoose.Schema({      name:{        type:String,        required:true,        unique:true      },      author:{        type:String,        default:'匿名'      },      style:{        type:String,        enum:['言情','推理','小说','城市','战争']      },      price:Number,      is\_hot:Boolean,      tags:Array,      pub\_time:Date    })    //创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查    let bookModel = mongoose.model("books",bookSchema)    //新增数据 ，使用bookModel的create()方法    //select,字段筛选,0表示不要的字段1表示要的字段      bookModel.find().select({price:1,is\_hot:1}).exec()      .then(data=>{              console.log(data);              mongoose.disconnect()          }).catch(err=>{            console.log(err);            mongoose.disconnect()            return          })    // //sort    //  bookModel.find().sort("price").exec()    //   .then(data=>{    //       console.log(data);    //       mongoose.disconnect()    //   }).catch(err=>{    //     console.log(err);    //     mongoose.disconnect()    //     return    //   })  })  db.on('close',()=>{    console.log("关闭数据库...");  }) |

### 效果

|  |
| --- |
| 数据库连接成功  [  {  \_id: new ObjectId('65dfafcb77c245bb92c6749c'),  price: 20.5,  is\_hot: true  },  {  \_id: new ObjectId('65dfb023f37501f1bf2f5f6d'),  price: 19.8,  is\_hot: true  },  {  \_id: new ObjectId('65dfbc6e10b2936a63de6201'),  price: 18.8,  is\_hot: true  },  {  \_id: new ObjectId('65dfbc6e10b2936a63de6202'),  price: 18.8,  is\_hot: true  },  {  \_id: new ObjectId('65dfbc6e10b2936a63de6203'),  price: 18.8,  is\_hot: true  },  {  \_id: new ObjectId('65dfbc6e10b2936a63de6204'),  price: 22.7,  is\_hot: true  },  {  \_id: new ObjectId('65dfdbdd122c223f0b4ca12c'),  price: 18.8,  is\_hot: true  },  {  \_id: new ObjectId('65dfdcad2bfa98da0d55d82f'),  price: 18.8,  is\_hot: true  },  {  \_id: new ObjectId('65e0b8301a477c02feacda9d'),  price: 17.5,  is\_hot: true  },  {  \_id: new ObjectId('65e0b8301a477c02feacda9e'),  price: 35.5,  is\_hot: true  },  {  \_id: new ObjectId('65e0b8301a477c02feacda9f'),  price: 45,  is\_hot: true  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa0'),  price: 30.8,  is\_hot: true  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa1'),  price: 25,  is\_hot: true  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa2'),  price: 21,  is\_hot: true  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa3'),  price: 25,  is\_hot: false  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa4'),  price: 10.8,  is\_hot: false  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa5'),  price: 9.8,  is\_hot: false  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa6'),  price: 12.8,  is\_hot: false  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa7'),  price: 22.8,  is\_hot: true  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa8'),  price: 25,  is\_hot: true  }  ]  关闭数据库... |

## 2）sort，排序1表示从小到大，-1表示从大到小，如我们想查询所有数据，然后安装price降序排列

|  |
| --- |
| //mongoose查询后处理，promise语法  let mongoose = require('mongoose')  mongoose.connect('mongodb://127.0.0.1/bookshop')    //设置数据库回调事件处理  var db = mongoose.connection;  db.on('error', console.error.bind(console, 'connection error:')); //连接失败  db.once('open', function() { //连接成功    // we're connected!    console.log("数据库连接成功");    //所有的业务代码都需要在这里写，不能写在外面      //先创建文档的结构对象Schema,添加Boolean类型    let bookSchema = new mongoose.Schema({      name:{        type:String,        required:true,        unique:true      },      author:{        type:String,        default:'匿名'      },      style:{        type:String,        enum:['言情','推理','小说','城市','战争']      },      price:Number,      is\_hot:Boolean,      tags:Array,      pub\_time:Date    })    //创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查    let bookModel = mongoose.model("books",bookSchema)    //新增数据 ，使用bookModel的create()方法    //select,字段筛选,0表示不要的字段1表示要的字段      // bookModel.find().select({price:1,is\_hot:1}).exec()      // .then(data=>{      //         console.log(data);      //         mongoose.disconnect()      //     }).catch(err=>{      //       console.log(err);      //       mongoose.disconnect()      //       return      //     })    // //sort,1表示从小到大，-1表示降序排列     bookModel.find().sort({price:-1}).exec()      .then(data=>{          console.log(data);          mongoose.disconnect()      }).catch(err=>{        console.log(err);        mongoose.disconnect()        return      })  })  db.on('close',()=>{    console.log("关闭数据库...");  }) |

### 效果

|  |
| --- |
| 数据库连接成功  [  {  \_id: new ObjectId('65e0b8301a477c02feacda9f'),  name: '富爸爸财务自由之路',  author: '罗伯特·清崎',  style: '理财',  price: 45,  is\_hot: true,  tags: [ '理财', '财商思维' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacda9e'),  name: '决定孩子一生幸福的财商教育',  author: '卫裕峰 ',  style: '理财',  price: 35.5,  is\_hot: true,  tags: [ '理财', '财商思维' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa0'),  name: '餐巾纸金融学：30秒内快速提高财商',  author: '蒂娜·海伊',  style: '理财',  price: 30.8,  is\_hot: true,  tags: [ '理财', '财商思维' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa1'),  name: '个体突围：真正的高手，都有破局思维',  author: '艾玛・加侬',  style: '思维',  price: 25,  is\_hot: true,  tags: [ '思维', '心计' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa3'),  name: '同桌的你',  author: '苏陌',  style: '言情',  price: 25,  is\_hot: false,  tags: [ '言情', '爱情小说' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa8'),  name: '谋杀的魅影',  author: '褚盟',  style: '推理',  price: 25,  is\_hot: true,  tags: [ '推理', '恐怖' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa7'),  name: '幸运数字3',  author: '玛格瑞.爱琳汉姆',  style: '推理',  price: 22.8,  is\_hot: true,  tags: [ '推理', '小说' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65dfbc6e10b2936a63de6204'),  name: '倚天屠龙记',  author: '曹雪芹',  style: '言情',  price: 22.7,  is\_hot: true,  tags: [ '神话传说', '武侠小说' ],  pub\_time: 2024-02-28T23:06:22.535Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa2'),  name: '破局\_打造人才供应链',  author: '许锋',  style: '思维',  price: 21,  is\_hot: true,  tags: [ '思维', '心计' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65dfafcb77c245bb92c6749c'),  name: '水浒传',  author: '施耐庵',  style: '城市',  price: 20.5,  is\_hot: true,  tags: [ '神话传说', '四大名著' ],  pub\_time: 2024-02-28T22:12:27.331Z,  \_\_v: 0  },  {  \_id: new ObjectId('65dfb023f37501f1bf2f5f6d'),  name: '三国演义',  author: '罗贯中',  style: '战争',  price: 19.8,  is\_hot: true,  tags: [ '神话传说', '四大名著' ],  pub\_time: 2024-02-28T22:13:55.453Z,  \_\_v: 0  },  {  \_id: new ObjectId('65dfbc6e10b2936a63de6201'),  name: '天龙八部',  author: '匿名',  style: '言情',  price: 18.8,  is\_hot: true,  tags: [ '神话传说', '武侠小说' ],  pub\_time: 2024-02-28T23:06:22.534Z,  \_\_v: 0  },  {  \_id: new ObjectId('65dfbc6e10b2936a63de6202'),  name: '射雕英雄传',  author: '匿名',  style: '言情',  price: 18.8,  is\_hot: true,  tags: [ '神话传说', '四大名著' ],  pub\_time: 2024-02-28T23:06:22.535Z,  \_\_v: 0  },  {  \_id: new ObjectId('65dfbc6e10b2936a63de6203'),  name: '金瓶梅',  author: '匿名',  style: '言情',  price: 18.8,  is\_hot: true,  tags: [ '神话传说', '四大禁书' ],  pub\_time: 2024-02-28T23:06:22.535Z,  \_\_v: 0  },  {  \_id: new ObjectId('65dfdbdd122c223f0b4ca12c'),  name: '西游记',  author: '吴承恩',  price: 18.8,  is\_hot: true,  tags: [ '神话传说', '四大名著' ],  pub\_time: 2024-02-29T01:20:29.199Z,  \_\_v: 0  },  {  \_id: new ObjectId('65dfdcad2bfa98da0d55d82f'),  name: '红楼梦',  author: '曹雪芹',  style: '言情',  price: 18.8,  is\_hot: true,  tags: [ '神话传说', '四大名著' ],  pub\_time: 2024-02-29T01:23:57.603Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacda9d'),  name: '富爸爸21世纪的生意',  author: '罗伯特·清崎',  style: '理财',  price: 17.5,  is\_hot: true,  tags: [ '理财', '财商思维' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa6'),  name: '月光森林',  author: '葵田谷',  style: '推理',  price: 12.8,  is\_hot: false,  tags: [ '推理', '小说' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa4'),  name: '初恋',  author: '库列绍夫',  style: '言情',  price: 10.8,  is\_hot: false,  tags: [ '言情', '爱情小说' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa5'),  name: '大诱拐',  author: '天藤真 ',  style: '推理',  price: 9.8,  is\_hot: false,  tags: [ '推理', '小说' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  }  ]  关闭数据库... |

## 3）截取，skip是跳过几条，limit是限定，如：查询所有数据然后跳过5条，然后按照price降序排列再取出前5条

|  |
| --- |
| //mongoose查询后处理，promise语法  let mongoose = require('mongoose')  mongoose.connect('mongodb://127.0.0.1/bookshop')    //设置数据库回调事件处理  var db = mongoose.connection;  db.on('error', console.error.bind(console, 'connection error:')); //连接失败  db.once('open', function() { //连接成功    // we're connected!    console.log("数据库连接成功");    //所有的业务代码都需要在这里写，不能写在外面      //先创建文档的结构对象Schema,添加Boolean类型    let bookSchema = new mongoose.Schema({      name:{        type:String,        required:true,        unique:true      },      author:{        type:String,        default:'匿名'      },      style:{        type:String,        enum:['言情','推理','小说','城市','战争']      },      price:Number,      is\_hot:Boolean,      tags:Array,      pub\_time:Date    })    //创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查    let bookModel = mongoose.model("books",bookSchema)    //新增数据 ，使用bookModel的create()方法    //select,字段筛选,0表示不要的字段1表示要的字段      // bookModel.find().select({price:1,is\_hot:1}).exec()      // .then(data=>{      //         console.log(data);      //         mongoose.disconnect()      //     }).catch(err=>{      //       console.log(err);      //       mongoose.disconnect()      //       return      //     })    // //sort,1表示从小到大，-1表示降序排列    //  bookModel.find().sort({price:-1}).exec()    //   .then(data=>{    //       console.log(data);    //       mongoose.disconnect()    //   }).catch(err=>{    //     console.log(err);    //     mongoose.disconnect()    //     return    //   })    //数据截取，skip跳过 limit限定    //查询所有数据然后跳过5条，然后按照price降序排列再取出前5条  //会先执行sort再执行skip，然后再执行limit    bookModel.find().skip(5).sort({price:-1}).limit(5).exec()      .then(data=>{          console.log(data);          mongoose.disconnect()      }).catch(err=>{        console.log(err);        mongoose.disconnect()        return      })  })  db.on('close',()=>{    console.log("关闭数据库...");  }) |

### 效果

|  |
| --- |
| 数据库连接成功  [  {  \_id: new ObjectId('65e0b8301a477c02feacdaa8'),  name: '谋杀的魅影',  author: '褚盟',  style: '推理',  price: 25,  is\_hot: true,  tags: [ '推理', '恐怖' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa7'),  name: '幸运数字3',  author: '玛格瑞.爱琳汉姆',  style: '推理',  price: 22.8,  is\_hot: true,  tags: [ '推理', '小说' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65dfbc6e10b2936a63de6204'),  name: '倚天屠龙记',  author: '曹雪芹',  style: '言情',  price: 22.7,  is\_hot: true,  tags: [ '神话传说', '武侠小说' ],  pub\_time: 2024-02-28T23:06:22.535Z,  \_\_v: 0  },  {  \_id: new ObjectId('65e0b8301a477c02feacdaa2'),  name: '破局\_打造人才供应链',  author: '许锋',  style: '思维',  price: 21,  is\_hot: true,  tags: [ '思维', '心计' ],  pub\_time: 2024-02-29T17:00:32.248Z,  \_\_v: 0  },  {  \_id: new ObjectId('65dfafcb77c245bb92c6749c'),  name: '水浒传',  author: '施耐庵',  style: '城市',  price: 20.5,  is\_hot: true,  tags: [ '神话传说', '四大名著' ],  pub\_time: 2024-02-28T22:12:27.331Z,  \_\_v: 0  }  ]  关闭数据库... |

# 11.mongoose代码模块化，

## 0.我们先调整一下代码结构，方便对比，我们新建一个文件夹basic-codes，把00-09的代码移动到里面，然后我们新建一个mongoose-modulize文件夹用来学习mongoose模块化开发

|  |
| --- |
|  |

## 1.我们来学习模块化开发，在mongoose-modulize文件夹里面新建一个index.js,把一些基本代码拷贝过来

|  |
| --- |
| //字段验证，使用promise写法  let mongoose = require('mongoose')  mongoose.connect('mongodb://127.0.0.1/bookshop')    //设置数据库回调事件处理  var db = mongoose.connection;  db.on('error', console.error.bind(console, 'connection error:')); //连接失败  db.once('open', function() { //连接成功    // we're connected!    console.log("数据库连接成功");    //所有的业务代码都需要在这里写，不能写在外面    //1.新增数据    //1.1先创建文档的结构对象Schema,添加Boolean类型    let bookSchema = new mongoose.Schema({      name:{        type:String,        required:true,        unique:true      },      author:{        type:String,        default:'匿名'      },      style:{        type:String,        enum:['言情','推理','小说','城市','战争']      },      price:Number,      is\_hot:Boolean,      tags:Array,      pub\_time:Date    })    //1.2创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查    let bookModel = mongoose.model("books",bookSchema)    //1.3新增数据 ，使用bookModel的create()方法          bookModel.create({              name:"红楼梦",              author:"曹雪芹",              style:'言情',              price:18.8,              is\_hot:true,              tags:['神话传说','四大名著'],              pub\_time:new Date()          }).then(data=>{          console.log(data);          mongoose.disconnect()          }).catch(err=>{              console.log(err);              return;          })  })  db.on('close',()=>{    console.log("关闭数据库...");  }) |

## 2.文件在mongoose-modulize文件夹里面新建一个db文件夹，然后在里面新建一个文件db.js,把index.js里面除了open数件响应函数以外的代码剪切到db.js,然后需要用一个函数把内容包装了再导出

### db/db.js

|  |
| --- |
| /\*\*   \*   \* @param {   \*   success :连接成功的回调函数   \*   error :连接失败的回调函数   \* }   \* @returns   \*/  module.exports = function(success,error) { //导出函数      let mongoose = require('mongoose')      mongoose.connect('mongodb://127.0.0.1/bookshop3')        //设置数据库回调事件处理      var db = mongoose.connection;      db.on('error', ()=>{          error()      }); //连接失败      db.once('open', ()=> { //连接成功         success()      })        db.on('close',()=>{          console.log("关闭数据库...");      })    } |

## 3.在index.js中导入这个函数，然后调用这个函数并且传递连接成功后和连接识别的回调函数，如我们在连接成功后马上插入一条数据

|  |
| --- |
| let mongoose = require('mongoose')  let db = require('./db/db') //导入函数  //使用我们封装的函数  db(()=>{       let bookSchema = new mongoose.Schema({          name:{          type:String,          required:true,          unique:true          },          author:{          type:String,          default:'匿名'          },          style:{          type:String,          enum:['言情','推理','小说','城市','战争']          },          price:Number,          is\_hot:Boolean,          tags:Array,          pub\_time:Date      })    //1.2创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查    let bookModel = mongoose.model("books",bookSchema)    //1.3新增数据 ，使用bookModel的create()方法          bookModel.create({              name:"红楼梦",              author:"曹雪芹",              style:'言情',              price:18.8,              is\_hot:true,              tags:['神话传说','四大名著'],              pub\_time:new Date()          }).then(data=>{           console.log(data);   mongoose.disconnect()          }).catch(err=>{              console.log(err);              return;          })    },    ()=>{      console.log("error...");  }) |

### 运行程序，效果如下

|  |
| --- |
|  |

### 打开命令行窗口，用命令查看，发现数据库创建成功并且新增了一条数据

|  |
| --- |
|  |

## 4.继续封装，在db目录所在的目录里面新建一个model文件夹，在里面新建一个bookModel.js文件，

### 把index.js中创建模型的代码剪切然后粘贴到bookModel.js中，然后在这个文件里面导入mongoose，并且把创建的模型导出

### bookModel.js

|  |
| --- |
| let mongoose = require('mongoose')  let bookSchema = new mongoose.Schema({      name:{      type:String,      required:true,      unique:true      },      author:{      type:String,      default:'匿名'      },      style:{      type:String,      enum:['言情','推理','小说','城市','战争']      },      price:Number,      is\_hot:Boolean,      tags:Array,      pub\_time:Date  })  //创建模型对象,bookModel就是对应books集合，她可以完成数据的增删改查  let bookModel = mongoose.model("books",bookSchema)  //导出模型对象  module.exports = bookModel |

### 然后在index.js里面导入模型，然后我们修改代码新增一条数据

|  |
| --- |
| let mongoose = require('mongoose')  //导入函数  let db = require('./db/db')  //导入模型  let bookModel = require('./model/bookModel')  //使用我们封装的函数  db(()=>{    //1.3新增数据 ，使用bookModel的create()方法          bookModel.create({              name:"水浒传",              author:"施耐庵",              style:'言情',              price:30,              is\_hot:true,              tags:['神话传说','四大名著'],              pub\_time:new Date()          }).then(data=>{           console.log(data);           mongoose.disconnect()          }).catch(err=>{              console.log(err);              return;          })    },    ()=>{      console.log("error...");  }) |

#### 运行程序成功，然后我们在命令行窗口里面窗口，发现数据是添加了进来

|  |
| --- |
|  |

### 以后我们需要创建新的集合，必须在model文件里面新建一个对应集合的model文件，在里面创建标签导出Model，如：我们新一个movies集合，如下图

|  |
| --- |
|  |

## 5.我们可以把上面的代码优化一下，使得它只需要传递一个成功的函数，需要修改db/db.js

|  |
| --- |
| /\*\*   \*   \* @param {   \*   success :连接成功的回调函数   \*   error :连接失败的回调函数   \* }   \* @returns   \*/  module.exports = function(success,error) { //导出函数      if(typeof error !=='function'){          error = ()=>{              console.log("数据库连接失败~~");          }      }      let mongoose = require('mongoose')      mongoose.connect('mongodb://127.0.0.1/bookshop3')        //设置数据库回调事件处理      var db = mongoose.connection;      db.on('error', ()=>{          console.log("连接失败");          return      }); //连接失败      db.once('open', ()=> { //连接成功         success()      })        db.on('close',()=>{          console.log("关闭数据库...");      })    } |

### 然后我们在index.js中就可以这么写

|  |
| --- |
| let mongoose = require('mongoose')  //导入函数  let db = require('./db/db')  //导入模型  let bookModel = require('./model/bookModel')  //使用我们封装的函数  db(()=>{    //1.3新增数据 ，使用bookModel的create()方法          bookModel.create({              name:"天龙八部",              author:"金庸",              style:'言情',              price:30,              is\_hot:true,              tags:['神话传说','四大名著'],              pub\_time:new Date()          }).then(data=>{           console.log(data);           mongoose.disconnect()          }).catch(err=>{              console.log(err);              return;          })    }) |

### 我们再新增一条数据，发现成功了

|  |
| --- |
|  |

### 从命令行查看：也是有的

|  |
| --- |
|  |

## 6.把数据库相关的内容做成一个配置文件，我们在db文件夹所在的目录新建一个config目录，然后在里面新建一个config.js,内容如下

|  |
| --- |
| module.exports = {      DBHOST:'127.0.0.1',      DBPORT:27017,      DBNAME:"bookshop3"  } |

## 7.修改db/db.js里面数据库的连接语句

|  |
| --- |
| /\*\*   \*   \* @param {   \*   success :连接成功的回调函数   \*   error :连接失败的回调函数   \* }   \* @returns   \*/  module.exports = function(success,error) { //导出函数      if(typeof error !=='function'){          error = ()=>{              console.log("数据库连接失败~~");          }      }      let mongoose = require('mongoose')      //导入配置文件      let {DBHOST,DBPORT,DBNAME} = require('../config/config')      let dbUrl = `mongodb://${DBHOST}:${DBPORT}/${DBNAME}`      mongoose.connect(dbUrl)        //设置数据库回调事件处理      var db = mongoose.connection;      db.on('error', ()=>{          console.log("连接失败");          return      }); //连接失败      db.once('open', ()=> { //连接成功         success()      })        db.on('close',()=>{          console.log("关闭数据库...");      })    } |

### 然后我们打开index.js，修改一下代码，插入一条数据，发现成功了

|  |
| --- |
| let mongoose = require('mongoose')  //导入函数  let db = require('./db/db')  //导入模型  let bookModel = require('./model/bookModel')  //使用我们封装的函数  db(()=>{    //1.3新增数据 ，使用bookModel的create()方法          bookModel.create({              name:"倚天屠龙记",              author:"金庸",              style:'言情',              price:35,              is\_hot:true,              tags:['神话传说','四大名著'],              pub\_time:new Date()          }).then(data=>{           console.log(data);           mongoose.disconnect()          }).catch(err=>{              console.log(err);              return;          })    }) |
|  |

### 以后如果有需要改数据库的路径或者文件，直接在config/config.js里面修改就好了，非常方便