# 数据库

数据库即存储数据的仓库，它是独立于Node.js之外的软件，可以通过API去操作它。

## MongoDB数据库安装

Node.js通常使用MongoDB作为其数据库，具有高性能，易使用，存储数据方便等特点，完全使用JavaScript语法即可操作。下载

## MongoDB可视化软件

MongoDB可视化操作软件，使用图形界面操作数据库的一种方式。下载

## Mongoose第三方包

使用Node.js操作MongoDB数据库需要依赖Node.js第三方包mongoose，使用`npm install mongoose`命令下载

## MongoDB概念

| 术语 | 概念 | 解释 |
| --- | --- | --- |
| database | 数据库 | mongoDB数据库软件中可以建立多个数据库 |
| collection | 集合 | 一组数据的集合，可以理解为JavaScript中的数组 |
| document | 文档 | 一条具体的数据，可以理解为JavaScript中的对象 |
| field | 字段 | 文档中的属性名称，可以理解为JavaScript中的对象属性 |

## 数据库操作

### 开启mongoDB服务

在命令行工具中运行`net start mongodb`即可开启MongoDB服务。

### 创建数据库

在MongoDB中不需要显式创建数据库，如果正在使用的数据库不存在，MongoDB会自动创建。

### 数据库连接

```
// 引用mongoose包
const mongoose = require('mongoose');
// 数据库链接
mongoose.connect('mongodb://localhost/playground')
    .then(() => console.log('数据库连接成功'))
    .catch(err => console.log('数据库连接失败', err));
```

### 创建集合

创建集合实际上就是对集合设定规则。

```
// 设置集合规则
const courseSchema = new mongoose.Schema({
    name: String,
    author: String,
    tags: [ String ],
    data: { type: Date, default: Date.now },
    isPublished: Boolean
});
// 使用规则创建集合 返回集合类(集合构造函数)
const Course = mongoose.model('Course', courseSchema);
```

**创建文档**

创建文档实际上就是向集合中插入具体的数据。

```
// 创建集合类的实例
const course = new Course({
    name: 'Node.js course',
    author: 'wangjian',
    tags: ['node', 'backend'],
    isPublished: true
});
// 保存实例
course.save();
```

插入数据的另一种形式

```
Course.create({name: 'JavaScript基础', author: 'jiely', isPublish: true}, (err, doc) => {
     //  错误对象
    console.log(err)
     //  当前插入的文档
    console.log(doc)
});
// create还支持promise 可以写成下面的形式
Course.create({name: 'JavaScript基础', author: 'jiely', isPublish: true})
      .then(doc => console.log(doc))
      .catch(err => console.log(err))
```

**查询文档**

```
Course.find({
    name: 'wangjian',
    isPublished: true
})
.limit(10),
.sort({name: 1}) // 1 升序 -1 降序
.select({name: 1, tags: 1})
.exec((err, data) => {})
```

### 删除文档

```
 // 删除单个
Course.findOneAndDelete({}).then(result => console.log(result))
// findOneAndDelete只会删除一个如果有多条数据 就删除第一个
```

```
 // 删除多个
User.deleteMany({}).then(result => console.log(result))
```

```
Course.findByIdAndRemove(id, err => {});
```

### 更新文档

```
// 更新单个
User.updateOne({查询条件}, {要修改的值}).then(result => console.log(result))
```

```
// 更新多个
User.updateMany({查询条件}, {要更改的值}).then(result => console.log(result))
```

```
// 根据id更新
Course.findByIdAndUpdate(id, {
    $set: {
        author: 'mosh',
        isPublished: false
    }
}, err => {})
```

### 多集合联合查询（集合关联）

通常不同集合的数据之间是有关系的，例如文章信息和用户信息存储在不同集合中，但文章是某个用户发表的，要查询文章的所有信息包括发表用户，就需要用到集合关联。

```
// 用户集合
const User = mongoose.model('User', new mongoose.Schema({ name: { type: String } }));
// 文章集合
const Post = mongoose.model('Post', new mongoose.Schema({
    title: { type: String },
    // 使用ID将文章集合和作者集合进行关联
    author: { type: mongoose.Schema.Types.ObjectId, ref: 'User' }
}));
//联合查询
Post.find()
      .populate('author')
      .then((err, result) => console.log(result));
```

### Mongoose验证

在创建集合规则时，可以设置当前字段的验证规则，验证失败就则输入插入失败。

常见的验证规则:

- required: true 必传字段
- minlength：3 字符串最小长度
- maxlength: 20 字符串最大长度
- min: 2 数值最小为2
- max: 100 数值最大为100
- enum: ['html', 'css', 'javascript', 'node.js']
- trim: true 去除字符串两边的空格
- validate: 自定义验证器
- default: 默认值

**在catch中获取错误信息**

```
Post.create({title:'aa', age: 60, category: 'java', author: 'bd'})
    .then(result => console.log(result))
    .catch(error => {
        // 获取错误信息对象
        const err = error.errors;
        // 循环错误信息对象
        for (var attr in err) {
            // 将错误信息打印到控制台中
            console.log(err[attr]['message']);
        }
    })
```

**用户列表案例**

1. 搭建网站服务器，实现客户端与服务器端的通信
2. 连接数据库，创建用户集合，向集合中插入文档
3. 当用户访问/list时，将所有用户信息查询出来
    1. 实现路由功能
    2. 呈现用户列表页面
    3. 从数据库中查询用户信息 将用户信息展示在列表中
4. 将用户信息和表格HTML进行拼接并将拼接结果响应回客户端
5. 当用户访问/add时，呈现表单页面，并实现添加用户信息功能
6. 当用户访问/modify时，呈现修改页面，并实现修改用户信息功能
    - 修改用户信息分为两大步骤
        1. 增加页面路由 呈现页面
        2. 在点击修改按钮的时候 将用户ID传递到当前页面
        3. 从数据库中查询当前用户信息 将用户信息展示到页面中
    - 实现用户修改功能
        1. 指定表单的提交地址以及请求方式
        2. 接受客户端传递过来的修改信息 找到用户 将用户信息更改为最新的

7. 当用户访问/delete时，实现用户删除功能