Nodejs全栈入门

2020年6月7日 23:09

项目：https://github.com/daohlm/imooc-todo-api2-20695-20200610

来自 <https://www.imooc.com/learn/1199>

alias 别名	expresses 表达；快递	middleware 中间件	extension 扩展；延期；广度	migration 迁移；迁徙；移动；徙动	

1-5.nodemon热部署插件

npm install modemon -D -D会装到devDependence

在 package.json 的 scripts 加上 start: "nodemon ./src/app. js"

1-6.nrm和npm介绍

nvm -管理node.js版本
为什么要进行node.js管理？
因为项目的不同，有可能node.js的版本不同，不同版本的node.js可能会影响到项目的启动
nvm ls -查看当前安装的node.js版本
nvm use node.js版本 -设置使用哪个版本
nvm install node.js版本 -安装node.js对应版本，没有指定则安装最新的

来自 <https://www.imooc.com/video/20684>

nrm 是用来管理npm源
作用：可快速切换npm源
可以自己新增源，方便公司npm私服使用

来自 <https://www.imooc.com/video/20684>

npm install nrm -g
nrm ls 列出配置源
nrm -h
nrm current
nrm use taobao
add <registry> <url> [home]

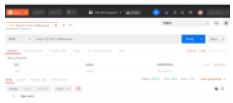
nvm ls 列出nodejs版本
nvm install [-s] <version>
mvn ls-remote --lts 查询所有nodejs已公开发布的版本
mvn install v10.14.1
mvn use v8.9.0 设置当前临时版本，重启应用后无效
mvn -v
mvn alias default v8.9.0 设置默认版本

2-1web应用基融和第一个 expressa应用

npm init
git init
根目录下加 .gitignore 文件

npm install express -S -D会装到dependence 生产环境
npm install modemon -D -D会装到devDependence 开发环境
package.json 的 script 加上 "start":"nodemon./src/app.js" 热部署
npm install express --save-dev --save-dev 是你开发时候依赖的东西，--save 是你发布之后还依赖的东西。

postman：



IDEA 开发遇挫
app.get 下划线
File | Settings | Languages & Frameworks | JavaScript | Libraries
勾上 Node.js Core 然后Download

```
const express = require('express');
const app = express();
// app.use((req,res) => {
//     res.json({
//         name: '张三'
//     })
// });
app.get('/name/:age', (req, res) => {
    let {age} = req.params;
    res.send({
        name: 'tom',
        age: age,
        query: req.query
    });
})
app.post('/name', (req, res) => {
    res.send('tompost');
})
app.all('/name', (req, res) => {
    res.send('tomall')
})
app.all('*', (req, res) => {
    res.send('all*')
})
app.listen(3000, () => {
    console.log('server启动成功');
});
//纯nodejs
// let http = require('http');
// let server = http.createServer(((req, res) => {
//     res.write("hello");
// }));
// server.listen(3000, '127.0.0.1', () => {
//     console.log('服务启动成功');
// });
```

2-5 express路由AP使用

const app = exoress(); app指的是 web服务的一个实例
app.use 使用中间件
const router = express.Router(); 他是 app的一个子对象

file app.js

```
let express = require('express');
let app = express();
const memberRouter =
require('./member.router');
app.use('/member', memberRouter);
app.listen(3000, () => {
    console.log('服务器启动成功');
});
```

file member.router.js

```
let express = require('express');
let router = express.Router();
router.get('/list', (req, res) => {
    res.json({
        list: [{id: 1, name: '李思思'}]
    })
});
module.exports = router;
```

2-6中间件

什么是express中间件　　类似插线板
内置中间件和第三方中间件介绍
自定义中间件

```
function demo_middleware(err, req, res, next) {
    // 1.异常
    // 2.处理下业务功能，然后转交控制权--next
```

1.app级别的使用
注册的时候，一定要在最顶部

```
function demo_middleware(err, req, res, next) {
    // 1.异常
    // 2.处理下业务功能，然后转交控制权--next
    // 3.响应请求--结束响应-->当作路由的处理函数
}
```

1.app级别的使用
注册的时候，一定要在最顶部
app.use-->api去加载进来
2.router级别
3.异常处理-->app级别-->router

```
const express = require('express');
const app = express();
function validNameMiddleware(req, res, next) {
  let {name} = req.query;
  if (!name || !name.length) {
    res.json({
      message: '缺少name参数'
    });
  } else {
    next();
  }
}
app.all('*', validNameMiddleware);
app.post('/name', (req, res) => {
  res.send('tom post');
})
app.listen(3000, () => {
  console.log('server启动成功');
});
```

加载一个内置的 static 中间件

```
app.use(express.static('static', {
    // 允许访问static目录下的静态资源
    extensions: ['html', 'htm']
}))
```

multipart中间件用于文件上传，常用

```
function validLoginParamMiddleware(req, res, next) {
    let {name, password} = req.query;
    if (!name || !password) {
        res.write('参数错误')
    } else {
        req.formdata = {name, password}
        next()
    }
}
router.get('/login', [validLoginParamMiddleware], (req, res) => {
    let {formdata} = req;
    res.write('')
})
```

2-7异常处理

```
const express = require('express');
const app = express();
app.get('/name', (req, res) => {
  throw new Error('异常');
})
function demoMiddleware(req,res,next) {
  try {
    // MySQL TODO
  } catch (e) {
    next(e)
  }
}
function errHandlerMiddleware(err, req, res, next) {
  if (err) {
    res.json({
      status: 1,
      errMessage: err.message
    })
  }
}
function notFoundHandlerMiddleware(req, res ,next) {
  res.status(500).json({
    errMessage: 'API不存在'
  })
}
app.use(errHandlerMiddleware);
app.use(notFoundHandlerMiddleware);
app.listen(3000, () => {
  console.log('server 启动成功');
});
```

2-9 sequel ize集成和使用

什么是ORM
Object Relational Mapping(对象关系映射)
Sequelize作用
在 nodes应用中集成 sequelize

```
npm install sequelize -S
npm install --save sequelize-cli 这是帮助使用的
npx sequelize-cli init 会在根目录创建几个目录
        config目录
        migrations 数据库迁移文件夹
        models ORM对象
            index.js 用它初始化ROM对象，看官方文档
        seeders 初始化脚本，建表初始化数据的脚本
npx sequelize-cli model:generate --name User --attributes name:string 创建User模型
npm install mysql2 -S 安装MySQL驱动
npx sequelize-cli db:migrate --env=development 迁移模型到数据库
```

```
const express = require('express');
const app = express();
const models = require('../models'); // 模型对象
// modules.user
app.get('/create', async (req, res) => {
  let {name} = req.query;
  // promise user --> sequelize 对象
  let user = await models.User.create({name});
  console.log(user);
  res.json({message: '创建成功', user})
})
app.get('/list', async (req, res) => {
  let list = await models.User.findAll();
  res.json({list});
})
app.get('/detail/:id', async (req, res) => {
  let {id} = req.params;
  let user = await models.User.findOne({where: {id}});
  res.json({user});
})
app.listen(3000, () => {
  console.log('server 启动成功');
});
```

3 使用express+mysql+sequelize实现任务管理项目

3-2 api设计

```
npm init -y
git init
npm install express mysql2 sequelize -S
npm install sequelize-cli -S
npm install nodemon -D
npm install body-parser -S 解析请求参数中间件
```

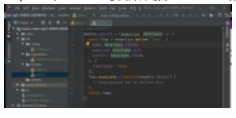
安装git
参考https://www.liaoxuefeng.com/wiki/896043488029600/896067074338496
git config --global user.name "daohlm" 设置git全局用户
git config --global user.email "xxx@xxx.xxx" 设置git全局邮箱
git remote add origin git@github.com:daohlm/imooc-todo-api2-20695-20200610.git 关联git仓库
https://juejin.im/post/5d123c4b5188252a5615b54c IDEA提交代码到GitHub

3-30RM模型创建

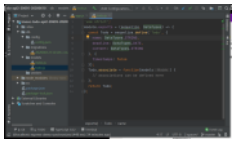
数据库的初始化
1.创建一个数据库
2.使用 `sequelize cli` 初始化 项目的数据库配置信息
    `npx sequelize init`
3.生成模型文件
1. migrate 文件

```
cd db
npx sequelize init
```

sequelize在create时会默认插入 createAt和updateAt 两个字段的数据，这时需要在 model 下 配置 timstamp:false



1.创建一个数据库
2.使用 `sequelize cli` 初始化 项目的数据库配置信息
`npx sequelize init`
3.生成模型文件
1. migrate 文件
2. model 文件
`npx sequelize model:generate --name Todo --attributes name:string,deadline:date,content:string`
4.持久化模型对应的数据库表(根据model文件往数据库生成表) `npx sequelize db:migrate`



3-6运维和发布

npm I pm2 -g
pm2 init
使用 pm2 start ecosystem.config.js 代替 npm start
pm2 list
pm2 restart 0
pm2 restart ecosystem.config.js
pm2 log
启动命令/运维命令/运维文档

4-1项目回顺

1.技术栈
1.node-->http,异常
2.web框架，express、hapi、koa、egg
3.参数校验
4.MySQL的使用
5.ORM，sequelize
2.技术的关键点
api
web->webserver->router->handler->orm->db
3.注意事项
1.需要详细的 模型设计 -> 模型之间的关系()
2.API的使用文档-->API文档生成(使用)工具
3.测试