# NODE的高级用法

## 模板引擎

第三方模块，用于代码拼接

## art-template模板引擎

#### 下载

npm install art-template

#### 基本使用

const template = require("art-template")

const path = require("path")

const pathFile = path.join(\_\_dirname, "index.art")

const html = template(pathFile, {

    user: [{

            name: "李白",

            age: 18

        },

        {

            name: "李白1",

            age: 181

        }

    ]

})

console.log(html)

<ul>

**{{each user}}**

<li>

**{{$value.name}}**

**{{$value.age}}**

</li>

**{{/each}}**

</ul>

#### 子模版

<ul>

**{{each user}}**

<li>

**{{$value.name}}**

**{{$value.age}}**

</li>

**{{/each}}**

</ul>

**{{include "./header.art"}} //这就是子模版的用法，直接引用其他的**

#### 模板继承

//模板

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>Document</title>

**{{block "link"}}{{/block}} //知识声明位置，和名字**

</head>

<body>

**{{block "content"}}{{/block}}**

</body>

</html>

**{{extend "./2.art"}} //声明模板位置**

**{{block "content"}} //填充具体内容**

<p>**{{msg}}**</p>

**{{/block}}**

**{{block "link"}}**

<link herf="lll">

**{{/block}}**

#### 5．模板变量

通过 npm i dateformat

const template = require("art-template")

const path = require("path")

const dateFormat = require("dateFormat")

template.defaults.imports.dateFormat = dateFormat //导入模板变量

const pathFile = path.join(\_\_dirname, "1.art")

const html = template(pathFile, {

    time: new Date()

})

console.log(html)

**{{dateFormat(time,"yyyy-mm-dd")}}**

#### 6.模板的一些基本设置

const template = require("art-template")

const path = require("path")

const dateFormat = require("dateFormat")

// const pathFile = path.join(\_\_dirname, "1.art")

template.defaults.imports.dateFormat = dateFormat //导入模板变量

template.defaults.root = path.join(\_\_dirname) //设置模板根目录

template.defaults.extname = ".art" //配置模板的默认后缀

const html = template("1", {

    time: new Date()

})

console.log(html)

#### 7.一些常用的第三方模块

router //路由

server-static //开放静态资源

# express框架

## express框架的特性

路由定义更简单

http请求参数的简化处理

对模板引擎支持度高，更容易渲染页面

提供了中间件，更易处理http请求

功能更好扩展

## express基础使用部分

const express = require("express")

//创建网站服务器

const app = express()

app.get("/", (req, res) => {

    res.send("hello")

})

app.get("/list", (req, res) => {

    res.send("<h1>你好</h1>")

})

app.listen(3000, () => {

    console.log("创建成功")

})

## 中间件

#### 中间件的一般用法

const express = require("express")

//创建网站服务器

const app = express()

app.get("/list", (req, res, next) => {

    req.name = "李白"

    next()

})

app.get("/list", (req, res) => {

    res.send(req.name)

})

app.listen(3000, () => {

    console.log("创建成功")

})

#### 中间件（use）

const express = require("express")

//创建网站服务器

const app = express()

app.use((req, res, next) => { //无论是哪种请求都会经过这，拥有预处理功能，必须写在前面

    console.log("这是use中间件")

    req.name = "张三"

    next()

})

app.get("/list", (req, res) => {

    res.send(req.name)

})

app.listen(3000, () => {

    console.log("创建成功")

})

#### 中间件的应用

##### 路由保护，可以在访问路有前，先判断是否登陆，设置拦截

const express = require("express")

//创建网站服务器

const app = express()

app.use("/list", (req, res, next) => { //判断登陆状态

    let isLogin = true

    if (isLogin) {

        next()

    } else {

        res.send("没登陆")

    }

})

app.get("/list", (req, res) => {

    res.send("hello")

})

app.listen(3000, () => {

    console.log("创建成功")

})

##### 拦截全部请求先做处理

const express = require("express")

//创建网站服务器

const app = express()

app.use((req, res, next) => {

  console.log("在维护")

})

app.get("/list", (req, res) => {

    res.send("hello")

})

app.listen(3000, () => {

    console.log("创建成功")

})

##### 自定义404页面

const express = require("express")

//创建网站服务器

const app = express()

app.get("/list", (req, res) => {

    res.send("hello")

})

app.use((req, res, next) => {

    res.status(404).send("404 没有该页面") //status是设置状态码的

})

app.listen(3000, () => {

    console.log("创建成功")

})

##### 处理错误的中间件(集中处理错误)

const express = require("express")

const fs = require("fs")

//创建网站服务器

const app = express()

app.get("/index", (req, res, next) => {

    fs.readFile("./11.txt", "utf-8", (err, data) => {

        if (err) {

            next(err)

        } else {

            res.send(data)

        }

    })

})

app.use((err, req, res, next) => {

    res.status(500).send(err.message)

})

app.listen(3000, () => {

    console.log("创建成功")

})

##### 捕获错误

const express = require("express")

const fs = require("fs")

const promisify = require("util").promisify

const readFile = promisify(fs.readFile)

//创建网站服务器

const app = express()

app.get("/index", async(req, res, next) => {

    try {

        await readFile("./2.txt")

    } catch (err) {

        next(err)

    }

})

app.use((err, req, res, next) => {

    res.status(200).send(err.message)

})

app.listen(3000, () => {

    console.log("创建成功")

})

##### 构建模块化路由

const express = require("express")

const app = express() //创建网站服务器

const login = require("./code/login")

const admin = require("./code/admin")

app.use("/login", login)

app.use("/admin", admin)

app.listen(3000, () => {

    console.log("开启成功")

})

const express = require("express")

const admin = express.Router() //创建路由

login.get("/index", (req, res) => {

    res.send("登陆页面")

})

module.exports = admin

## 使用get方法完成表单提交

表单提交注意必须要有name=“”

app.get("/pinlun",function(req,res){

    var list1=req.query

    list1.id=list.length

    list.unshift(list1) //添加到list

    //实现重定向，即成功后会跳到根目录

   res.redirect("/")

})

## 使用post方法完成表单提交

**安装插件 npm i --save body-parser**

var bodyParser=require("body-parser") //引入

//配置body-parser插件

app.use(bodyParser.urlencoded({extended:false}))

app.use(bodyParser.json()) //可以没有

//使用post方法完成表单提交

app.post("/pinlun",function(req,res){

var list1=req.body

list1.id=list.length

list.unshift(list1)

res.redirect("/")

})

## express路由参数

const express = require("express")

const app = express()

app.get("/index/:id/:name", (req, res) => {

    res.send(req.params)

})

app.listen(3000)

## 静态资源的处理

const express = require("express")

const path = require("path")

const app = express()

    //第一个参数表示必须以此开头，第二个表示开放的路径

app.use("/public", express.static(path.join(\_\_dirname, "public")))

app.get("/index", (req, res) => {

    res.send()

})

app.listen(3000)

## 在express中使用art-template模板引擎

下载

**npm i --save art-template express-art-template**

引入

//第一个参数是模板的后缀

app.engine("html",require("express-art-template"))

app.set() //设置模板所在的位置，默认是当前目录平级的views文件夹

使用：

app.get("/",function(req,res){

//res.render("模板文件的路径"，模板数据)

   res.render("index.html",{

        list:list

    })

})

var list=[

    {

        id:0,

        name:"李白",

        message:"我是李大白0"

    },

]