# 以前我们响应用户的特定页面的请求都是用fs读取文件然后返回给用户，其实在nodejs中有模板引擎，我们可以比较高效的渲染页面，常见的有jade、ejs和art-template，今天我们来学习art-template

# 1.安装art-template和express-art-template

|  |
| --- |
|  |

# 2.在项目中配置art-template

|  |
| --- |
|  |

# 3. .art-template模板引擎语法

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 输出 **标准语法**   |  | | --- | | {{value}} {{data.key}} {{data['key']}} {{a ? b : c}} {{a || b}} {{a + b}} |   **原始语法**   |  | | --- | | <%= value %> <%= data.key %> <%= data['key'] %> <%= a ? b : c %> <%= a || b %> <%= a + b %> |   模板一级特殊变量可以使用 $data 加下标的方式访问：   |  | | --- | | {{$data['user list']}} |  原文输出 **标准语法**   |  | | --- | | {{@ value }} |   **原始语法**   |  | | --- | | <%- value %> |   原文输出语句不会对 HTML 内容进行转义处理，可能存在安全风险，请谨慎使用。 条件 **标准语法**   |  | | --- | | {{if value}} ... {{/if}} {{if v1}} ... {{else if v2}} ... {{/if}} |   **原始语法**   |  | | --- | | <% if (value) { %> ... <% } %> <% if (v1) { %> ... <% } else if (v2) { %> ... <% } %> |  循环 **标准语法**   |  | | --- | | {{each target}}  {{$index}} {{$value}} {{/each}} |   **原始语法**   |  | | --- | | <% for(var i = 0; i < target.length; i++){ %>  <%= i %> <%= target[i] %> <% } %> |  1. target 支持 array 与 object 的迭代，其默认值为 $data。 2. $value 与 $index 可以自定义：{{each target val key}}。  变量 **标准语法**   |  | | --- | | {{set temp = data.sub.content}} |   **原始语法**   |  | | --- | | <% var temp = data.sub.content; %> |  模板继承 **标准语法**   |  | | --- | | {{extend './layout.art'}} {{block 'head'}} ... {{/block}} |   **原始语法**   |  | | --- | | <% extend('./layout.art') %> <% block('head', function(){ %> ... <% }) %> |   模板继承允许你构建一个包含你站点共同元素的基本模板“骨架”。范例：   |  | | --- | | <!--layout.art--> <!doctype html> <html> <head>  <meta charset="utf-8">  <title>{{block 'title'}}My Site{{/block}}</title>   {{block 'head'}}  <link rel="stylesheet" href="main.css">  {{/block}} </head> <body>  {{block 'content'}}{{/block}} </body> </html> |  |  | | --- | | <!--index.art--> {{extend './layout.art'}}  {{block 'title'}}{{title}}{{/block}}  {{block 'head'}}  <link rel="stylesheet" href="custom.css"> {{/block}}  {{block 'content'}} <p>This is just an awesome page.</p> {{/block}} |   渲染 index.art 后，将自动应用布局骨架。 子模板 **标准语法**   |  | | --- | | {{include './header.art'}} {{include './header.art' data}} |   **原始语法**   |  | | --- | | <% include('./header.art') %> <% include('./header.art', data) %> |  1. data 数默认值为 $data；标准语法不支持声明 object 与 array，只支持引用变量，而原始语法不受限制。 2. art-template 内建 HTML 压缩器，请避免书写 HTML 非正常闭合的子模板，否则开启压缩后标签可能会被意外“优化。  过滤器 **注册过滤器**   |  | | --- | | template.defaults.imports.dateFormat = function(date, format){/\*[code..]\*/}; template.defaults.imports.timestamp = function(value){return value \* 1000}; |   过滤器函数第一个参数接受目标值。  **标准语法**   |  | | --- | | {{date | timestamp | dateFormat 'yyyy-MM-dd hh:mm:ss'}} |   {{value | filter}} 过滤器语法类似管道操作符，它的上一个输出作为下一个输入。  **原始语法**   |  | | --- | | <%= $imports.dateFormat($imports.timestamp(date), 'yyyy-MM-dd hh:mm:ss') %> |   如果想修改 {{ }} 与 <% %>，请参考 [解析规则](https://aui.github.io/art-template/zh-cn/docs/rules.html)。 |

## 注意：我们一般使用标准语法

# 官方示例

|  |
| --- |
|  |

## 注意：建议把模板文件的后缀名设置为html，因为vscode编写html有提示

# 4.art-template使用练习

# 1.准备工作：

## 0）把express\_hello项目复制一份，改名：express\_hello\_art-template，

## 1）在项目中新建一个views文件夹，用来存放模板，把一些不需要的文件删除，然后把一些文件使得修改

### login.html

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>User Login</title>  </head>  <body>      <h3>用户登录</h3>      <form action="/login" method="post">          <p>用户姓名：<input type="text" name="username" placeholder="输入用户名"></p>          <p>用户密码：<input type="password" name="password"></p>          <p><input type="submit" value="注册"></p>      </form>  </body>  </html> |

### register.html

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>User Register</title>  </head>  <body>      <h3>用户注册</h3>      <form action="/register" method="post">          <p>用户姓名：<input type="text" name="username" placeholder="输入用户名"></p>          <p>用户邮箱：<input type="text" name="email" placeholder="输入邮箱"></p>          <p>用户密码：<input type="password" name="password"></p>          <p>确认密码：<input type="password" name="confirm"></p>          <p><input type="submit" value="注册"></p>      </form>  </body>  </html> |

### 01-server-art.js

|  |
| --- |
| //express art-template的使用  let express = require('express')  let bodyParser = require('body-parser')  let app = express()  let PORT = 3003  //模拟登录用数据  let USER = 'admin'  let PASS = '12345'  app.get('/',(req,res)=>{      res.send('<h1>Welcome to index</h1>')  })  //配置静态服务器  app.use(express.static('public'))  //配置404  app.get('\*',(req,res)=>{ //如果我们没有为两个路径配置路由，就会被他匹配到      res.send('<h1>Not Found</h1>')  })  //post方法  //1.需要利用bodyParser来生成一个能够解析表单数据的中间件  let urlParser = bodyParser.urlencoded({extended:false}) //false接收字符串或者数组，true接收任意类型  // //2.全局注册解析表单数据的中间件  app.use(urlParser)  app.post('/register',(req,res)=>{      // console.log(req.body);      let {username,email,password} = req.body      console.log(username,email,password);        //业务逻辑：      //校验数据，是否为空，是否符合规定      //查询数据库看看用户名是否已经被注册了      //...      //假设数据已经通过上面的校验了，用户成功注册，现在需要跳转到登录页面      res.redirect('/login.html')//路由重定向  })  app.post('/login',(req,res)=>{//模拟登录处理      // console.log(req.body);      let {username,password} = req.body      console.log(username,password);      if(username != USER || password != PASS){          res.send(`登录失败，用户名或者密码错误！！！`)          return      }      res.send(`<h4>登录成功，欢迎 ${username}</h4>`)  })  app.listen(PORT,()=>{      console.log(`url: http://localhost:${PORT}/`);  }) |

## 2）打开终端，定位到改项目的路径

|  |
| --- |
|  |

## 3）安装插件：art-template 和express-art-template，使用命令yarn add art-template express-art-template

|  |
| --- |
|  |

# 2.项目练习

## 1）先测试一下，项目是可以正常运行的

|  |
| --- |
|  |
|  |

## 2）在views文件夹里面新建一个index.html，内容如下

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>    <meta charset="UTF-8">    <meta name="viewport" content="width=device-width, initial-scale=1.0">    <title>首页</title>  </head>  <body>    {{if data }}    <h1>首页</h1>    <hr />    <ul>      <!-- 写法1 -->      <!--老语法：each object as value index 新语法：each object value index  -->      <!-- {{each data item i}}        <li>          <h2>{{i+1}}.<a href={{item.href}}>{{item.name}}</a></h2>        </li>      {{/each}} -->      <!-- 写法2 -->      {{each data}}      <li>        <h2>{{$index +1}}.<a href={{$value.href}}>{{$value.name}}</a></h2>      </li>      {{/each}}    </ul>    {{else}}    <h1>首页</h1>    {{/if}}  </body>  </html> |

### 注意art-template的语法，each语句有2种写法

## 3）然后我们在服务器文件中设置art-template，一共有4个步骤，

|  |
| --- |
| //express art-template的使用  let path = require('path')  let express = require('express')  let bodyParser = require('body-parser')  let app = express()  let PORT = 3003  //模拟登录用数据  let USER = 'admin'  let PASS = '12345'  //设置art-template引擎,4个步骤  //1.引入art-template,使用对应的引擎  app.engine('html',require('express-art-template'))  //2.设置项目环境  app.set('view options',{      debug:process.env.NODE\_ENV!== "production"  })  //3.设置视图路径，也就是到哪里找模板文件  app.set('views',path.join(\_\_dirname,'views'))  //4.设置模板文件的后缀名  app.set('view engine','html')  //配置静态服务器  app.use(express.static('public'))  app.get('/',(req,res)=>{      let data=[          {name:'登录',href:"/login.html"},          {name:'注册',href:"/register.html"},      ]      res.render('index',{data}) //注意：传递数据给模板需要用花括号，否则拿不到数据  })  //配置404  app.get('\*',(req,res)=>{ //如果我们没有为两个路径配置路由，就会被他匹配到      res.send('<h1>Not Found</h1>')  })  //post方法  //1.需要利用bodyParser来生成一个能够解析表单数据的中间件  let urlParser = bodyParser.urlencoded({extended:false}) //false接收字符串或者数组，true接收任意类型  // //2.全局注册解析表单数据的中间件  app.use(urlParser)  app.post('/register',(req,res)=>{      // console.log(req.body);      let {username,email,password} = req.body      console.log(username,email,password);        //业务逻辑：      //校验数据，是否为空，是否符合规定      //查询数据库看看用户名是否已经被注册了      //...      //假设数据已经通过上面的校验了，用户成功注册，现在需要跳转到登录页面      res.redirect('/login.html')//路由重定向  })  app.post('/login',(req,res)=>{//模拟登录处理      // console.log(req.body);      let {username,password} = req.body      console.log(username,password);      if(username != USER || password != PASS){          res.send(`登录失败，用户名或者密码错误！！！`)          return      }      res.send(`<h4>登录成功，欢迎 ${username}</h4>`)  })  app.listen(PORT,()=>{      console.log(`url: http://localhost:${PORT}/`);  }) |

## 测试一下：

|  |
| --- |
|  |

## 点击任意一个链接可以进入对应的页面，如点击登录，进入登录界面

|  |
| --- |
|  |

# art-template其实还是蛮简单的