

8.2 vue-server-renderer

- 1.初次使用服务器渲染
- 2.把生成的字符串发给客户端
- 3.读取模板文件
- 4.在模板中使用外部的数据
- 5.同构应用源码结构
- 6.webpack构建配置

1.初次使用服务器渲染

- 新建工程vue-server
- 安装vue 和 vue-server-renderer
- 新建server.js

```
1 const Vue = require('vue')
2 const renderer = require('vue-server-renderer').createRenderer()
3 const app = new Vue({
4   template: `
5     <div id="app">
6       <h1>{{ message }}</h1>
7     </div>
8   `,
9   data:{
10     message: '测试数据'
11   }
12 })
13
14 renderer.renderToString(app, (err,html) => {
15   if (err) {
16     throw err
17   }
18   console.log(html);
19 })
```

```
1 $: node server.js
2 <div id="app" data-server-rendered="true"><h1>测试数据</h1></div>
```

- data-server-rendered="true" 是用来将来客户端接管的入口

2.把生成的字符串发给客户端

- 安装npm i express
- 使用express返回html

```
1 const Vue = require('vue')
2 const renderer = require('vue-server-renderer').createRenderer()
3 const express = require('express')
4 // 创建server实例
5 const server = express()
6 // 添加路由
7 server.get('/', (req, res) => {
8   const app = new Vue({
9     template: `
10       <div id="app">
11         <h1>{{ message }}</h1>
12       </div>
13     `,
14     data: {
15       message: '测试数据'
16     }
17   })
18   renderer.renderToString(app, (err, html) => {
19     if (err) {
20       // 出现错误, 返回500的异常
21       res.status(500).end('Internal Server Error.')
22     }
23     // 方式一: 设置编码
24     // res.setHeader('Content-Type', 'text/html; charset=utf8')
25     // res.end(html)
26
27     // 方式二: 返回html文档
28     res.setHeader('Content-Type', 'text/html; charset=utf8')
29     res.end(`
```

```

30     <!DOCTYPE html>
31     <html lang="en">
32     <head>
33         <meta charset="UTF-8">
34         <meta name="viewport" content="width=device-width, initial-scale=1.0">
35         <title>Document</title>
36     </head>
37     <body>
38         ${html}
39     </body>
40 </html>
41 `)
42 })
43
44 })
45
46 server.listen(3000)

```

3. 读取模板文件

- 新建index.template.html(注释一定要写)

```

1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5     <meta charset="UTF-8">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7     <title>Document</title>
8 </head>
9
10 <body>
11     <!--vue-ssr-outlet-->
12 </body>
13
14 </html>

```

- createRenderer中指定模板

```
1 const Vue = require('vue')
2 const fs = require('fs')
3 const renderer = require('vue-server-renderer').createRenderer({
4   // 指定模板
5   template: fs.readFileSync('./index.template.html','utf-8')
6 })
7 const express = require('express')
8 // 创建server实例
9 const server = express()
10 // 添加路由
11 server.get('/', (req,res) => {
12   const app = new Vue({
13     template: `
14       <div id="app">
15         <h1>{{ message }}</h1>
16       </div>
17     `,
18     data:{
19       message: '测试数据'
20     }
21   })
22   renderer.renderToString(app, (err,html) => {
23     if (err) {
24       // 出现错误,返回500的异常
25       res.status(500).end('Internal Server Error.')
26     }
27
28     // 方式三: renderer指定模板
29     res.setHeader('Content-Type','text/html;charset=utf8')
30     res.end(html)
31   })
32
33 })
34
35 server.listen(3000)
```

4.在模板中使用外部的数据

- 在renderToString中传入数据

```
1 const Vue = require('vue')
2 const fs = require('fs')
3 const renderer = require('vue-server-renderer').createRenderer({
4   // 指定模板
5   template: fs.readFileSync('./index.template.html', 'utf-8')
6 })
7 const express = require('express')
8 // 创建server实例
9 const server = express()
10 // 添加路由
11 server.get('/', (req, res) => {
12   const app = new Vue({
13     template: `
14       <div id="app">
15         <h1>{{ message }}</h1>
16       </div>
17     `,
18     data: {
19       message: '测试数据'
20     }
21   })
22   renderer.renderToString(
23     app,
24     // 需要传入模板中的数据
25     {
26       title: '模板标题',
27       meta: `<meta name="description" content="测试数据">`
28     },
29     (err, html) => {
30       if (err) {
31         // 出现错误, 返回500的异常
32         res.status(500).end('Internal Server Error.')
33       }
34       res.setHeader('Content-Type', 'text/html; charset=utf8')
35       res.end(html)
```

```

36     }
37   )
38 })
39
40 server.listen(3000)

```

- 在html模板中使用 两个大括号渲染字符串

```

1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scal
   e=1.0">
7   <title>{{ title }}</title>
8 </head>
9
10 <body>
11   <!--vue-ssr-outlet-->
12 </body>
13
14 </html>

```

- 已html的方式渲染数据,三个大括号渲染html

```

1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta name="viewport" content="width=device-width, initial-scal
   e=1.0">
7   {{{ meta }}}
8   <title>{{ title }}</title>
9 </head>
10
11 <body>

```

```
12   <!--vue-ssr-outlet-->
13 </body>
14
15 </html>
```

5.同构应用源码结构

<https://ssr.vuejs.org/zh/guide/structure.html>

6.wepack构建配置