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.wepack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

```
const Vue = require('vue')
const renderer = require('vue-server-renderer').createRenderer()
const app = new Vue({
  template: `
    <div id="app">
      <h1>{{ message }}</h1>
    </div>
  `,
  data:{
    message: '测试数据'
  }
})

renderer.renderToString(app, (err,html) => {
  if (err) {
    throw err
  }
  console.log(html);
})
```

```
$: node server.js
<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

```
const Vue = require('vue')
const renderer = require('vue-server-renderer').createRenderer()
const express = require('express')
// 创建server实例
const server = express()
// 添加路由
server.get('/', (req,res) => {
  const app = new Vue({
    template: `
      <div id="app">
        <h1>{{ message }}</h1>
      </div>
    `,
    data:{
      message: '测试数据'
    }
  })
  renderer.renderToString(app, (err,html) => {
    if (err) {
      // 出现错误,返回500的异常
      res.status(500).end('Internal Server Error.')
    }
    // 方式一: 设置编码
    // res.setHeader('Content-Type','text/html;charset=utf8')
    // res.end(html)

    // 方式二: 返回html文档
    res.setHeader('Content-Type','text/html;charset=utf8')
    res.end(`
```

```
      <!DOCTYPE html>
      <html lang="en">
      <head>
          <meta charset="UTF-8">
          <meta name="viewport" content="width=device-width, initial-scale=1.0">
          <title>Document</title>
      </head>
      <body>
        ${html}
      </body>
      </html>
    `)
  })

})

server.listen(3000)
```

3.读取模板文件

- 新建index.template.html(注释一定要写)

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>

<body>
  <!--vue-ssr-outlet-->
</body>

</html>
```

- createRenderer中指定模板

```
const Vue = require('vue')
const fs = require('fs')
const renderer = require('vue-server-renderer').createRenderer({
  // 指定模板
  template: fs.readFileSync('./index.template.html','utf-8')
})
const express = require('express')
// 创建server实例
const server = express()
// 添加路由
server.get('/', (req,res) => {
  const app = new Vue({
    template: `
      <div id="app">
        <h1>{{ message }}</h1>
      </div>
    `,
    data:{
      message: '测试数据'
    }
  })
  renderer.renderToString(app, (err,html) => {
    if (err) {
      // 出现错误,返回500的异常
      res.status(500).end('Internal Server Error.')
    }

    // 方式三: renderer指定模板
    res.setHeader('Content-Type','text/html;charset=utf8')
    res.end(html)
  })

})

server.listen(3000)
```

4.在模板中使用外部的数据

- 在renderToString中传入数据

```
const Vue = require('vue')
const fs = require('fs')
const renderer = require('vue-server-renderer').createRenderer({
  // 指定模板
  template: fs.readFileSync('./index.template.html', 'utf-8')
})
const express = require('express')
// 创建server实例
const server = express()
// 添加路由
server.get('/', (req, res) => {
  const app = new Vue({
    template: `
      <div id="app">
        <h1>{{ message }}</h1>
      </div>
    `,
    data: {
      message: '测试数据'
    }
  })
  renderer.renderToString(
    app,
    // 需要传入模板中的数据
    {
      title: '模板标题',
      meta: `<meta name="description"content="测试数据">`
    },
    (err, html) => {
      if (err) {
        // 出现错误,返回500的异常
        res.status(500).end('Internal Server Error.')
      }
      res.setHeader('Content-Type', 'text/html;charset=utf8')
      res.end(html)
```

```
    }
  )
})

server.listen(3000)
```

- 在html模板中使用 两个大括号渲染字符串

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>{{ title }}</title>
</head>

<body>
  <!--vue-ssr-outlet-->
</body>

</html>
```

- 已html的方式渲染数据,三个大括号渲染html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  {{{ meta }}}
  <title>{{ title }}</title>
</head>

<body>
```

```
  <!--vue-ssr-outlet-->
</body>

</html>
```

5.同构应用源码结构

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

6.wepack构建配置