正课:

1. SPA
2. Vue-router
3. 脚手架
4. SPA: 单页面应用

什么是: 整个应用程序只有一个完整的HTML页面

每个“页面”其实仅是一段HTML片段/组件

所谓的”页面跳转”，只是将HTML中特定区域替换为新组件的HTML片段

为什么: 多页面 vs 单页面

|  |  |
| --- | --- |
| 多页面 | 单页面 |
| 项目中包含多个独立的完整HTML页面  缺: 每次跳转都向服务器重新请求页面，增加了请求的次数 | 整个项目只有一个完整的HTML页面，其它页面只是一段HTML片段/组件  优: 每次跳转不需要向服务器发送请求，减少了请求次数  缺: 不利于搜索引擎优化 |
| 页面跳转: 删除整棵DOM树，重新下载HTML，并重建整棵DOM树  缺: 效率低 | 页面跳转: 仅替换当前DOM树中的局部内容为新HTML片段，无需重建DOM树和等待下载  优: 效率高！ |
| 如果两个页面都要用到同一个资源，必须反复请求下载 | 如果两个“页面”都要用到同一个资源 |

1. Vue-router:

什么是: vue中专门实现单页面应用的路由器组件

何时: 只要在vue中实现单页面应用，都用vue-router

如何:

SPA应用的原理:

1. 先定义一个完整的HTML，其中包含多个<template>
2. 将<template>封装为子组件对象
3. 定义路由字典:

路由: 一个相对路径+一个组件

路由字典: 包含多对儿路径与组件对应关系的集合

如何: var routes=[

{path:”/”,component:home}, //默认首页

{path:”/home”,component:home},

{path:”/login”,component:login},

{path:”/\*”,component:{template:”<h1>404:xxx</h1>”}}

];

1. 将路由字典装入新创建的路由器对象中:

var router=new VueRouter({ routes })

1. 将路由器对象加入new Vue中:

new Vue({ el:”#app”, router })

1. 在<div id=”app”>中添加<router-view>占位标签

运行时: 地址栏中: index.html#/相对路径

router可监控#/相对路径的变化，并自动替换<router-view>为指定的组件片段.

嵌套路由:

什么是: 在一级路径下，又细分未多个二级路径

何时: 当多个”页面”/组件中包含部分相同的页面结构时

如何:

1. 定义一个父级组件，包含相同部分的结构和一个<router-view>
2. 在路由列表中:

var routes=[

{path:”/一级路径”,component:父组件,children:[

{path:”/一级路径/二级路径”, component:子组件},

… …

]}

]

运行时:

当访问路径以一级路径开头时: 就先用父组件代替div#app中的<router-view>

当一级路径之后，跟上二级路径: 再用子组件代替父组件中的<router-view>

路由跳转: 2种:

1. 在HTML中写死:

<router-link to=”/login”>文本</router-link>

在运行时，会自动翻译为<a href=”#/login”>文本

1. 在js中用程序跳转:

this.$router.push(“/新地址”)

//Vue模拟history的方式进行跳转

路由传参: 2步:

1. 在路由列表中定义参数:

{path:”/路径/:参数名”,component:组件}

强调: 一旦定义参数，则必须带参才能访问当前路径

1. 在跳转时，传参:

在HTML中: <router-link to=”/路径/参数值”>

强调: 跳转时，不要加:

在程序中:

1. 在下一个组件的程序中获得参数值:

this.$route.params.参数名

问题: 在带参数的“页面”，仅修改地址栏中的参数值，不会重新绑定页面内容

解决: 添加watch监听地址栏$route:

watch:{//监控

"$route":{//当地址栏中的内容发生变化时

handler(to){//自动调用handler

//to: 新的$route对象

this.lid=to.params.lid;

},

immediate:true//立刻

}

}

3. 脚手架:

什么是: 半成品项目，已经封装了大部分重复性劳动，只需要添加个性化功能即可。

为什么: 1. 节约代码量; 2. 统一项目结构

如何:

1. 安装脚手架命令行工具: vue-cli 用于在本地生成脚手架代码

cnpm i –g @vue/cli

1. 用vue-cli命令行工具在本地硬盘生成脚手架项目源代码

打开命令行cmd，进入想要创建项目的父级文件夹下:

执行: vue create 项目名/项目文件夹名

第一步: 提示: 使用默认npm下载会很慢，是否使用taobao镜像: Y或n都行

第二步: 提示: 使用默认配置，还是手动设置配置:   
 选: Manually select features

第三步: 选择组件: 按空格键选中或取消选中

选: Babel: 翻译工具: 将es6甚至更高版本的js翻译为es5的代码

Router: vue-router

Vuex: 下个老师讲

第四步: Use history mode for router:

默认模式: index.html#/相对路径

History模式: index.html/相对路径——需要服务器端配置重定向

选: n 不使用history

第五步: 将配置分别保存在多个配置文件中，还是集中保存在一个package.json中

选: In package.json

第六步: 是否将本次配置保存为今后项目的默认配置: 选 n

1. 安装完成后:

cd 项目目录

运行: npm run serve

提示: 编译成功

浏览: <http://localhost:8080>看到示例页面

脚手架的项目结构:

public: 保存不需要编译，就可直接浏览的静态资源:

比如：图片，再比如，公用的css，公用的js

同时还保存着整个项目唯一的完整的HTML页面

index.html

src: 保存着项目的所有源代码

assets: 专门用于保存，需要被打包的静态资源: css,图片或js

App.vue: 保存着根组件的<template>和根组件的css

main.js: 保存着new Vue()，监视着App.vue中的<div id=”app”>,以及对Vue的配置

运行时: App.vue和main.js 共同替换index.html中的<div id=”app”>

views: 专门保存“页面组件”的文件夹，将来有几个“页面”，views中，就有几个.vue文件

components: 专门保存各个页面中的子组件的文件夹

router.js: 专门定义路由器和路由列表的模块

每个.vue文件: 都是一个独立的组件，拥有专属的HTML，CSS，JS和数据

每个.vue文件中包含三部分:

<template>

<div>

组件的HTML片段  
 </div>

</template>

<script>

//引入子组件模块: @=”src/”

//相当于: var 组件对象名=require(“xxx.vue”)

Import 组件对象名 from “@/components/子组件.vue”

//将当前组件抛出为模块:

export default {

data(){

return { 模型变量 }

},

//同Vue中的Vue.component或new Vue()

//定义子组件

componets:{

子组件对象名

}

}

</script>

<style scoped>/\*仅限于当前组件范围内可用的样式\*/

当前组件的<template>专用的Css样式

.fade{ … }

</style>

安装axios: npm i –s axios

配置axios:

main.js中: import axios from 'axios'

Vue.prototype.axios=axios;

Xxx.vue中: 使用axios:

this.axios.get/post(…)