Vue 项目架构之路由详解，工具类封装（基于 jQuery 的 Ajax）

讲解之前，咱们接地气的描述一下单页面应用程序。

所谓单页面应用程序，其实就是只有一个页面，这个页面中有一个div，而后的页面，我们称为模板（或者组件），在切换页面的时候，改变这个div中的模板（或者组件）即实现的单页面应用程序。

本文讲解，均是在vue脚手架搭建vue框架完备之后：（脚手架使用顺序）

```
    npm install -g vue-cli
    vue init webpack XXXX
    cd XXXX
    npm install  /  npm install --
registry=https://registry.npm.taobao.org
    npm run dev
```


这个时候，最基本的vue框架就搭建完成了，基本目录解构如下：

- build 最终发布的代码存放位置。
- config 配置目录，包括端口号等。我们初学可以使用默认的。
- node_modules npm 加载的项目依赖模块
- src 这里是我们要开发的目录，基本上要做的事情都在这个目录里。里面包含了几个目录及文件：
- assets 放置一些图片，如logo等。
- components 目录里面放了一个组件文件，可以不用。
- App.vue 项目入口文件，我们也可以直接将组件写这里，而不使用 components 目

接下来看看router文件，在router文件夹下有一个index.js，这里需要注意，如果router文件夹下是index.js在引用的时候，import router from ‘./router’，如果router文件夹下不是index.js（比如：router.js），引用的时候import router from ‘./router/router’，我们在components目录下建立一个init.vue,home.vue,login.vue三个组件router/index.js。

```
import Vue from 'vue'
import VueRouter from 'vue-router'
//import store from '../store'

Vue.use(VueRouter)
  /**
   *  @path: 路径
   *  @name: 路径的别名
   *  @meta: 这个里面可以存放很多东西，比如下面这个：children
   *  @children: 子路由    比如：我们移动项目的header和footer，可以把这个组件写在init(名字随意)，init就是父路由，在父录音中包含的所以路由都是子路由，比如下面的home,可有有很多的父子路由，更具需求随意添加。这快细说，有很多东西，不明白的小伙伴可以私下@我
   */
   meta: {Title: '账户流水', Header: true, Back: true, Replace: false, Footer: false, requireAuth:true}
 /**
  * @Title:页面的标题和Header部分的标题
  * @Header:是否显示Header
  * @Back:是否显示返回按钮
  * @Replace:返回事件的参数（默认为go（-1））
  * @Footer:是否显示footer
  * @requireAuth:是否需要登陆
  */
consot router = new VueRouter({
    routes:[{
        path: '/init',
        name:'init',
        meta:{},
        component: resolve => require(['../components/init.vue'], resolve),
```

```
})
//修改网页的title
router.afterEach((transition) => {})
//路由拦截器的操作以及store的操作（总之一切路由加载操作都可以在这里面完成）
router.beforeEach((to, from, next) => {})
//暴露出口（vue中必须这么写，才能在外部被调用）
export default router
.....................................................................
.....................................................................
..............................
main.js

import Vue from 'vue'
//import Vuex from 'vuex'
//import FastClick from 'fastclick'
import router from './router'
//import store from './store'
import App from './App'
.......................
new Vue({
 // store,
  router,
  render: h => h(App)
}).$mount('#app')
new Vue({
  store,
  router,
  render: h => h(App)
}).$mount('#app')
```


好啦，一完整的路由配置就完成啦！每次添加页面，只需要在路由文件中添加相应的配置即可比如，init/login，home是init的子路由。

接下来我们谈谈工具类的封装这块。在src下新建一个util或者kit（名字不重要）。因为之前有很多小伙伴问我在vue中使用jQuery的Ajax(我是非常非常的不推荐在vue中使用jQuery的，但是为了响应号召，这里就提供一份儿基于jQuery封装的Ajax)。要用到

```
var webpack = require("webpack")

function resolve (dir) {...}

let webpackConfig = {
  entry: {
    app: './src/main.js'
  },
  output: {"path": config.build.assetsRoot...},
  resolve: {
    extensions: ['.js', '.vue', '.json'],
    alias: {
      'vue$': 'vue/dist/vue.esm.js',
      '@': resolve('src'),
      'jquery': 'jquery'
    }
  },
  module: {...},
  plugins: [
    new webpack.optimize.CommonsChunkPlugin('common.js'),
    new webpack.ProvidePlugin({
      jQuery: "jquery",
      $: "jquery"
    })
  ]
}
```

好啦，你可以在全局使用jQuery了。

kit/Request.js

```
import Vue from 'vue'
import store from '../store'
import router from '../router'

let Export = {}
let vue = new Vue({router})
Export.post = function (options) {
```

```
const Request = function (options) {
    let newDef = $.Deferred()
    store.dispatch('showloader')
    /**
     * @url：请求的接口地址
     * @method: 请求方式（GET/POST/PUT/DELETE）
     * @param: 请求参数{key:val} (id:'11',name:'name')
     * @headers: 请求的headers{key:val} (token:aabbccdd)
     * @useCache: 缓存（针对GET方式）
     * @skipValidation：跳过验证
     */
    let url = vue.SERVER_NAME + vue.API_PREFIX + options.url
    let method = options.method
    let param = options.data || true
    let headers = options.headers || true
    let useCache = options.cache || true
    let skipValidation = options.skipValidation || true
    if (method !== "GET") {
      param = (typeof param === "string") ? param :
JSON.stringify(param)
    }
    $.ajax({
      url: url,
      type: method.toUpperCase(),
      dataType: "json",
      contentType: "application/json; charset=utf-8",
      headers: headers,
      data: param,
      cache: !!useCache,
      success: (data) => {
        if (skipValidation) {
          store.dispatch('hideloader')
          newDef.resolve(data)
        } else if (handleApiResponseStatus(url, data)) {
          store.dispatch('showloader')
          newDef.resolve(data)
        }
      }
```

```
        if (data.codeText == "RESULT_LOGIN_EXPIRED") {
          console.info('哎呦喂！登陆超时')
          return false
        } else if (data.codeText == "RESULT_NEED_ADVANCE_AUTH") {
          console.info('哎呦喂！登陆超时,重新登陆')
        } else if (data.codeText == "RESULT_NEED_BINDPHONE") {
          return true
        } else if (data.codeText == "FORBIDDEN") {
          console.info('哎呦喂！权限验证失败')
          return false
        }
        return true
      }
      /**
       * 处理HTTP相应状态
       */
      function handleHttpResponseStatus(url, status) {
        console.info('handle Http Response Status Error: ' +
status)
        let statu = Number(status)
        if (statu == 404) {
          console.info('哎呦喂！我找不到页面')
        } else if (statu >= 500) {
          console.info('哎呦喂！服务器异常')
        } else {
          console.info('哎呦喂！网络出现异常')
        }
      }
      return newDef.promise()
}
```

在组件调用：

```
import Request from '../kit/Request'
Request.get({
      url: '',
```