# 目录结构

|  |
| --- |
|  |

# 生命周期

|  |
| --- |
| https://img2018.cnblogs.com/blog/1158910/201908/1158910-20190818204215037-2038459873.png  **红框内的是Nuxt的生命周期(运行在服务端)，**  **黄框内同时运行在服务端&&客户端上，**  **绿框内则运行在客户端**  红框、黄框内的周期都不存在Window对象  export default {   middleware(*context*){  console.log(this) // undefined       console.log(window) // 服务端报错    },    validate({*params*,*query*}){      console.log(this) // undefault      console.log(window) // 服务端报错      return true;    },    asyncData() {  console.log(this) // undefined      console.log(window) // 服务端报错    },    fetch() {  console.log(this) // 执向组件      console.log(window) // 服务端报错    },    created () {  console.log(this) // 执向组件      console.log(window) // undefined    },    mounted () {  console.log(this) // 执向组件      console.log(window) // Window {postMessage: ƒ, blur: ƒ, focus: ƒ, close: ƒ, frames: Window, …}    }  } |

## nuxtServerInit

|  |
| --- |
| 适用场景是对 store 操作  store/index.js  export *const* actions = {    nuxtServerInit(*store*, *context*) {      // 通常初始化数据到store当中      console.log('store：',*store*)      console.log('context：',*context*)    }  } |

## middleware中间件

|  |
| --- |
| 中间件执行流程顺序：nuxt.config.js->匹配布局->匹配页面 |

### nuxt.config.js配置

|  |
| --- |
| nuxt.config.js  router:{    middleware:'auth'  },  middleware/auth.js  export default (*context*)*=>*{    // context 服务端上下文    // 全局守卫业务    console.log('middleware：', *context*)  }  或  export default ({*store*,*route*,*redirect*,*params*,*query*,*req*,*res*})*=>*{    // context 服务端上下文    // 全局守卫业务    console.log('middleware：', *store*)  } |

### layouts布局页面配置

|  |
| --- |
| middleware/auth.js  export default (*context*)*=>*{    console.log('布局页面层级中间件：', *context*)  }  =======================================  layouts/default.vue  <script>  export default {    // middleware:'auth', 布局页面层级中间件定义  // 或    middleware(*context*) {      console.log('布局页面层级中间件：', context)    }  }  </script> |

### 普通页面配置

|  |
| --- |
| Pages/index.vue  在当前页面实例化之前执行  // SSR  // middleware:'auth'  // 或  middleware(*context*){  console.log('middleware pages：', context)  }, |

## validate

|  |
| --- |
| 参数校验，校验失败，则自动跳转到错误页面  Pages/index.vue    //参数的有效性    validate({*params*,*query*}) {      //校验业务      console.log('validate')      return true;    }, |

## asyncData

|  |
| --- |
| data(){      return {        a:1      }    },  //读数据，返回给组件    asyncData(*context*){      //异步业务逻辑，读取服务端数据      console.log('asyncData')      return {        b:2      }    }, |

## fetch

|  |
| --- |
| //读数据 , vuex    fetch({*store*}){      //异步业务逻辑，读取服务端数据提交给vuex      console.log('fetch')    }, |

## beforeCreate/ created

|  |
| --- |
| //SSR && CSR 有机会拿到window,拿不到时undefined    beforeCreate(){      console.log('beforeCreate')    },    created(){      console.log('created',this) // this指向当前组件    }, |

## beforeMount/ mounted

|  |
| --- |
| //CSR   window this指向组件    beforeMount(){},    mounted(){      console.log('mounted')    }, |

## beforeUpdate/ updated

|  |
| --- |
| beforeUpdate(){},    updated(){      console.log('updated')    }, |

## beforeDestroy/ destroyed

|  |
| --- |
| beforeDestroy(){},    destroyed(){      console.log('destroyed')    }, |

## activated/ deactivated

|  |
| --- |
| // 服务端渲染 激活、失活 不存在    activated(){},    deactivated(){}, |

# 路由

|  |
| --- |
|  |

# 路由守卫

## 前置

### 全局前置守卫

|  |
| --- |
| 第一种：    router:{  middleware:'auth',  }  Middleware/auth.js  export default ({*store*,*route*,*redirect*,*params*,*query*,*req*,*res*})*=>*{    // context 服务端上下文    // 全局守卫前置业务    // store 状态树信息    // route 一条目标路由信息    // redirect 强制跳转    // params,query 校验参数合理性    console.log('middleware nuxt.config outside 全局守卫前置业务')    redirect('/login')  }  =========================================  第二种:      middleware({*store*,*route*,*redirect*,*params*,*query*}){      //context 服务端上下文      // 全局守卫前置业务      // store 状态树信息      // route 一条目标路由信息      // redirect 强制跳转      // params,query 校验参数合理性      console.log('middleware layouts 全局守卫前置业务')      redirect('/reg')    }, |

### 组件独享前置守卫

|  |
| --- |
| pages/index.vue    middleware({*store*,*route*,*redirect*,*params*,*query*}){      //context 服务端上下文      // 组件独享前置守卫      // store 状态树信息      // route 一条目标路由信息      // redirect 强制跳转      // params,query 校验参数合理性      console.log('middleware pages 组件独享前置守卫')      redirect('/login')    } |

### 插件全局前置守卫

|  |
| --- |
| nuxt.config.js   plugins: [      '~/plugins/router',    ],  plugins/ router.js  export default ({*app*,*redirect*,*params*,*query*,*store*}) *=>* {    // app == vue实例    // redirect 跳转函数 √  *app*.router.beforeEach((*to*,*from*,*next*) *=>* {      // 全局前置的守卫， 插件      // next() next(true) next(false)      // next('/login') 不能跳转      // console.log('插件配置，全局前置',to)      //  next()     if(*to*.name=='login' || *to*.name=='reg'){        next()      }else {        redirect({name:'login'})      }    })  *app*.router.afterEach((*to*,*from*)*=>*{      console.log('插件全局后置守卫')    })  } |

## 后置

### 组件独享后置守卫

|  |
| --- |
| pages/index.vue  beforeRouteLeave(*to*,*from*,*next*){  *let* bl=window.confirm('是否要离开');     next(bl)  } |

### 插件全局后置守卫

|  |
| --- |
| nuxt.config.js   plugins: [      '~/plugins/router',    ],  plugins/ router.js  export default ({*app*,*redirect*,*params*,*query*,*store*}) *=>* {  *app*.router.afterEach((*to*,*from*)*=>*{      console.log('插件全局后置守卫')    })  } |

# 参数检验-错误路由定制

|  |
| --- |
| <nuxt-link to="/goods/1?a=1&b=2">商品01</nuxt-link>  <nuxt-link :to="{name:'goods-id',params:{id:2},query:{a:11,b:22}}">商品02</nuxt-link>  <nuxt-link :to="{name:'goods-id',params:{id:3},query:{a:111,b:222}}">商品03</nuxt-link>    name:'goods-detail',    validate({*params*,*query*}){      return typeof params.id === 'number'    },  layouts/error.vue  <template>    <div>      <h1 v-if="error.statusCode">{{error.message}}</h1>      <h1 v-else>应用发送异常</h1>      <button @click="$router.replace('/index')">跳转到首页</button>    </div>  </template>  <script>  export default {    //接受错误信息   error: {statusCode,message}    props:['error']  }  </script> |

# 统一动效和独享动效

|  |
| --- |
| asset/css/ transition.css  /\*  路由统一动效 \*/  /\* 动画形式 \*/  .page-enter-active, .page-leave-active{  *transition*:  opacity .5s;  }  /\* 入 退 \*/  .page-enter, .page-leave-active{  *opacity*:  0;  }  /\* 单路由动画 ->组件内部 transition: '类名' \*/  .test-enter-active, .test-leave-active{  *transition*:  .5s ease all;  }   .test-enter, .test-leave-active{  *margin-left*: -1000px;  }  nuxt.config.js  /\*   \*\* 全局样式   \*/  css: [    'assets/css/transition.css'  ],  ============================================================  独享动效  export default {    name:'goods-detail',    transition:'test' //'动画名' 可以使用全局样式test 或 当前组件样式test  }  </script>  <style scoped>    .test-enter-active, .test-leave-active{  *transition*:  .5s ease all;    }     .test-enter, .test-leave-active{  *margin-left*: -1000px;    }  </style> |

# 数据交互

|  |
| --- |
| 安装：@nuxtjs/axios、@nuxtjs/proxy  nuxtjs自带的模块都要在modules配置  next.config.js  modules: [     '@nuxtjs/axios'  ],  ------------------  static\data\list.json  {  *"title"*:"nuxt数据"  }  ------------------  pages\index.vue  async asyncData({*$axios*}){  *let* res = await $axios({url:'/data/list.json'})     console.log('读取到的静态资源',res.data)     return {        title:res.data.title,     }  },  async fetch({*$axios*}){  *let* res = await $axios({url:'/data/list.json'})    console.log('res',res.data.title)  }, |

# 跨域

|  |
| --- |
| nuxt.config.js  axios:{    proxy:true,    // 开启axios跨域    prefix:'/api', // baseUrl  },  proxy: {    '/api/':{      target:'http://localhost:3001',//代理转发的地址      changeOrigin:true,      pathRewrite:{        '^/api': '' // 把api开头替换为空      }    }  },  pages\index.vue  async asyncData({*$axios*}){    //读取跨域数据  *let* res2 = await $axios({url:'/api/goods/home',params:{\_limit:1}})    console.log('读取到的跨域资源',res2.data)    return {      data:res2.data.data    }  }, |

# 拦截器配置和token携带

|  |
| --- |
| nuxt.config.js  plugins: [     {       src:'~/plugins/axios',       ssr:true//服务端     }  ],  plugins\axios.js  export default *function*({*$axios*,*redirect*,*route*,*store*}) {    //基本配置  *$axios*.defaults.timeout=10000;    //请求拦截  *$axios*.onRequest(*config=>*{      console.log('请求拦截')  *config*.headers.token='eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImFsZXgiLCJfaWQiOiI1ZThhMGQ2MzczNDg2MDIzYTRmZDY4ZGYiLCJpYXQiOjE1ODY2MDIwODcsImV4cCI6MTU4NjY4ODQ4N30.bgzJALn6DiKMQuhohPjjml3cgjZbtXiU2Kfg9t1lcGo';      return *config*;    })    //响应拦截  *$axios*.onResponse(*res=>*{      // console.log('响应拦截')      if(*res*.data.err === 2 && *route*.fullPath !== '/login'){        redirect('/login?path='+*route*.fullPath)      }      return *res*    })    //错误处理  *$axios*.onError(*error=>*{      //处理      return *error*;    })  } |

# token携带

|  |
| --- |
|  |

# 状态管理

|  |
| --- |
|  |

# UI库服务端渲染

|  |
| --- |
|  |

# UI库服务端渲染

|  |
| --- |
|  |

# 全局定义

|  |
| --- |
|  |

# Meta优化

|  |
| --- |
|  |

# Sass、scss在服务端使用

|  |
| --- |
|  |

# 定义模板、模块、插件

|  |
| --- |
|  |

# 资源配置

|  |
| --- |
|  |

# 部署

|  |
| --- |
|  |

# Nuxt for ts

|  |
| --- |
|  |

# nuxt.config.js配置

## mode

|  |
| --- |
| 设置渲染模式   * 'spa': 没有服务器端渲染（只有客户端路由导航等） * 'universal': 默认，同构应用程序（服务器端呈现+客户端路由导航等） |