## Vue-Router导航守卫

Vue Router 是 [Vue.js](http://cn.vuejs.org/) 官方的路由管理器。它和 Vue.js 的核心深度集成，让构建单页面应用变得易如反掌。包含的功能有：

* 嵌套的路由/视图表
* 模块化的、基于组件的路由配置
* 路由参数、查询、通配符
* 基于 Vue.js 过渡系统的视图过渡效果
* 细粒度的导航控制
* 带有自动激活的 CSS class 的链接
* HTML5 历史模式或 hash 模式，在 IE9 中自动降级
* 自定义的滚动条行为

“导航”表示路由正在发生改变

有的时候，我们需要通过路由来进行一些操作，比如最常见的登录权限验证，当用户满足条件时，才让其进入导航，否则就取消跳转，并跳到登录页面让其登录。为此我们有很多种方法可以植入路由的导航过程：**全局的, 单个路由独享的, 或者组件级的**,

### 全局守卫

vue-router全局有三个守卫：

1. router.beforeEach 全局前置守卫 进入路由之前
2. router.beforeResolve 全局解析守卫(2.5.0+) 在beforeRouteEnter调用之后调用
3. router.afterEach 全局后置钩子 进入路由之后

**使用方法**：

// main.js 入口文件

import router from './router'; // 引入路由

router.beforeEach((to, from, next) => {

  next();

});

router.beforeResolve((to, from, next) => {

  next();

});

router.afterEach((to, from) => {

  console.log('afterEach 全局后置钩子');

});

to,from,next 这三个参数：

to和from是**将要进入和将要离开的路由对象**,路由对象指的是平时通过this.$route获取到的路由对象。

**next:Function** 这个参数是个函数，且**必须调用，否则不能进入路由**(页面空白)。

**入路由**(页面空白)。

* next() 进入该路由。
* next(false): 取消进入路由，url地址重置为from路由地址(也就是将要离开的路由地址)。
* next 跳转新路由，当前的导航被中断，重新开始一个新的导航。

我们可以这样跳转：next('path地址') 或者 next({path:''}) 或者 next({name:''})且允许设置诸如 replace: true、name: 'home' 之类的选项以及任何用在 [router-link 的 to prop](https://router.vuejs.org/zh/api/#to) 或 [router.push](https://router.vuejs.org/zh/api/#router-push) 中的选项。

### 路由独享守卫

如果你不想全局配置守卫的话，你可以为某些路由单独配置守卫：

const router = new VueRouter({

  routes: [

    {

      path: '/foo',

      component: Foo,

      beforeEnter: (to, from, next) => {

        // 参数用法什么的都一样,调用顺序在全局前置守卫后面，所以不会被全局守卫覆盖

        // ...  next();

      }

    }

  ]

})

### 路由组件内的守卫：

1. beforeRouteEnter 进入路由前
2. beforeRouteUpdate (2.2) 路由复用同一个组件时
3. beforeRouteLeave 离开当前路由时

const Foo = {

template: `...`,

beforeRouteEnter(to, from, next) {

// 在渲染该组件的对应路由被 confirm 前调用

// 不！能！获取组件实例 `this`

// 因为当守卫执行前，组件实例还没被创建

},

beforeRouteUpdate(to, from, next) {

// 在当前路由改变，但是该组件被复用时调用

// 举例来说，对于一个带有动态参数的路径 /foo/:id，在 /foo/1 和 /foo/2 之间跳转的时候，

// 由于会渲染同样的 Foo 组件，因此组件实例会被复用。而这个钩子就会在这个情况下被调用。

// 可以访问组件实例 `this`

},

beforeRouteLeave(to, from, next) {

// 导航离开该组件的对应路由时调用

// 可以访问组件实例 `this`

}

}

beforeRouteEnter 守卫 **不能** 访问 this，因为守卫在导航确认前被调用,因此即将登场的新组件还没被创建。

不过，你可以通过传一个回调给 next来访问组件实例。在导航被确认的时候执行回调，并且把组件实例作为回调方法的参数

beforeRouteEnter (to, from, next) {

console.log('在路由独享守卫后调用');

  next(vm => {

    // 通过 `vm` 访问组件实例`this` 执行回调的时机在mounted后面，

  })

}

注意 beforeRouteEnter 是支持给 next 传递回调的唯一守卫。对于 beforeRouteUpdate 和 beforeRouteLeave 来说，this 已经可用了，所以**不支持**传递回调，因为没有必要了。

