Vue2.0 探索之路——vue-router入门教程和总结

前言

这是关于vue的第三篇博文。
没想到写的还有人看，正是因为你们的阅读和点赞收藏，才给了我无比的动力。请关注我的专栏，我不会停更的。

最近也一直在想，前端知识怎么提高，前端知识的碎片化，让我感觉好多好多都不会，觉得这个时候我应该确定一个方向，重点的培养自己的招牌技能，再加以辅助技能。不过看了很多博文也暂时没有形成具体的路线，就先暂定写博文吧。

今天主要讲解一下 `vue-router` 的相关知识，路由路由嘛，在单页应用里还是蛮重要的。

安装和引入

首先我们先安装依赖
```
npm install vue-router
```

紧接着项目引入，看下面的图噢，非常清晰，代码就自己敲吧。

router.js 的配置

首先引入 `vue-router`组件，`Vue.use`是用来加载全局组件的。那下面我们就开始看看这个`VueRouter`的写法和配置吧。

mode:

默认为`hash`，但是用`hash`模式的话，页面地址会变成被加个#号比较难看了， `http://localhost:8080/#/linkParams/xuxiao`
所以一般我们会采用 `history`模式，它会使得我们的地址像平常一

样。http://localhost:8080/linkParams/xuxiao

base
应用的基路径。例如，如果整个单页应用服务在 /app/ 下，然后 base 就应该设为 "/app/"。
一般写成 __dirname，在webpack中有配置。

routes
routes就是我们的大核心了，里面包含我们所有的页面配置。
path 很简单，就是我们的访问这个页面的路径
name 给这个页面路径定义一个名字，当在页面进行跳转的时候也可以用名字跳转，要唯一哟
component 组件，就是咱们在最上面引入的 import ...了，当然这个组件的写法还有一种**懒加载**

懒加载的方式，我们就不需要再用import去引入组件了，直接如下即可。懒加载的好处是当你访问到这个页面的时候才会去加载相关资源，这样的话能提高页面的访问速度。

```
component: resolve => require(['./page/linkParamsQuestion.vue'], resolve)
```

router的使用

对于vue-router的使用，详细的可以看看文档，但是你知道的，文档也只是一个指引，具体的实现还是得靠自己码代码哟。不过我把官方文档放在下面，有兴趣的可以去看看。

> http://router.vuejs.org/zh-cn...

我通读文档 + 代码实现再结合平时项目开发的使用情况，主要讲下面几个点。

router传参数

在我们的平时开发跳转里，很明显，传参数是必要的。那么在vue-router中

如何跳转，如何传参数呢。请看下面。

1.路由匹配参数

首先在路由配置文件router.js中做好配置。标红出就是对 /linkParams/的路径做拦截，这种类型的链接后面的内容会被vue-router映射成name参数。

代码中获取name的方式如下：

```
let name = this.$route.params.name; // 链接里的name被封装进了 this.$route.para
```

2.Get请求传参

这个明明实在不好形容啊。不过真的是和Get请求一样。你完全可以在链接后加上?进行传参。

样例：http://localhost:8080/linkParamsQuestion?age=18

项目里获取：

```
let age = this.$route.query.age;   //问号后面参数会被封装进 this.$route.query;
```

编程式导航

这里就开始利用vue-router讲发起跳转了。其实也非常简单，主要利用<router-link>来创建可跳转链接，还可以在方法里利用this.$router.push('xxx')来进行跳转。

样例：<router-link to="/linkParams/xuxiao">点我不会怀孕</router-link>
上面的这个router-link就相当于加了个可跳转属性。

至于this.$router.push这里直接用官网的荔枝了

```
// 字符串,这里的字符串是路径path匹配噢，不是router配置里的name
this.$router.push('home')
```

```
// 对象
this.$router.push({ path: 'home' })

// 命名的路由 这里会变成 /user/123
this.$router.push({ name: 'user', params: { userId: 123 }})

// 带查询参数，变成 /register?plan=private
this.$router.push({ path: 'register', query: { plan: 'private' }})
```

导航钩子

导航钩子函数，主要是在导航跳转的时候做一些操作，比如可以做**登录的拦截**，而钩子函数根据其生效的范围可以分为 全局钩子函数、路由独享钩子函数和组件内钩子函数。

全局钩子函数

可以直接在路由配置文件router.js里编写代码逻辑。可以做一些全局性的路由拦截。

```
router.beforeEach((to, from, next)=>{
  //do something
  next();
});
router.afterEach((to, from, next) => {
    console.log(to.path);
});
```

每个钩子方法接收三个参数：

- **to: Route:** 即将要进入的目标 路由对象
- **from: Route:** 当前导航正要离开的路由
- **next: Function:** 一定要调用该方法来 resolve 这个钩子。执行效果依赖 next 方法的调用参数。
- **next()**: 进行管道中的下一个钩子。如果全部钩子执行完了，则导航的状态就是 confirmed （确认的）。

- next(false): 中断当前的导航。如果浏览器的 URL 改变了（可能是用户手动或者浏览器后退按钮），那么 URL 地址会重置到 from 路由对应的地址。

- next('/') 或者 next({ path: '/' }): 跳转到一个不同的地址。当前的导航被中断，然后进行一个新的导航。

确保要调用 next 方法，否则钩子就不会被 resolved。

路由独享钩子函数

可以做一些单个路由的跳转拦截。在配置文件编写代码即可

```
const router = new VueRouter({
  routes: [
    {
      path: '/foo',
      component: Foo,
      beforeEnter: (to, from, next) => {
        // ...
      }
    }
  ]
})
```

组件内钩子函数

更细粒度的路由拦截，只针对一个进入某一个组件的拦截。

```
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`
    // 因为当钩子执行前，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
    // 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`
  }
}
```

钩子函数使用场景

其实路由钩子函数在项目开发中用的并不是非常多，一般用于登录态的校验，没有登录跳转到登录页；权限的校验等等。当然随着项目的开发进展，也会有更多的功能可能用钩子函数实现会更好，我们知道有钩子函数这个好东西就行了，下次遇到问题脑海就能浮现，噢，这个功能用钩子实现会比较棒。我们的目的就达到了。当然，有兴趣的可以再去研究下源码的实现。开启下脑洞。

其他知识点

滚动行为

在利用vue-router去做跳转的时候，到了新页面如果对页面的滚动条位置有要求的话，可以利用下面这个方法。

```
const router = new VueRouter({
  routes: [...],
  scrollBehavior (to, from, savedPosition) {
    // return 期望滚动到哪个的位置
  }
})
```

scrollBehavior 方法接收 to 和 from 路由对象。
第三个参数 savedPosition 当且仅当 popstate 导航 (mode为 history 通过浏览器的 前进/后退 按钮触发) 时才可用。
这里就不细致的讲了，文档都有也非常简单，记住有这个东西就行。

```
//所有路由新页面滚动到顶部：
scrollBehavior (to, from, savedPosition) {
  return { x: 0, y: 0 }
}

//如果有锚点
```

```
scrollBehavior (to, from, savedPosition) {
  if (to.hash) {
    return {
      selector: to.hash
    }
  }
}
```

路由元信息

这个概念非常简单，就是在路由配置里有个属性叫 meta，它的数据结构是一个对象。你可以放一些key-value进去，方便在钩子函数执行的时候用。举个例子，你要配置哪几个页面需要登录的时候，你可以在meta中加入一个requiresAuth标志位。

```
const router = new VueRouter({
  routes: [
    {
      path: '/foo',
      component: Foo,
      meta: { requiresAuth: true }
    }
  ]
})
```

然后在 全局钩子函数 beforeEach中去校验目标页面是否需要登录。

```
router.beforeEach((to, from, next) => {
  if (to.matched.some(record => record.meta.requiresAuth)) {
    //校验这个目标页面是否需要登录
    if (!auth.loggedIn()) {
      next({
        path: '/login',
        query: { redirect: to.fullPath }
      })
    } else {
      next()
    }
  } else {
    next() // 确保一定要调用 next()
  }
})
```

> 这个**auth.loggedIn** 方法是外部引入的，你可以先写好一个校验是否登录的方法，再import进 router.js中去判断。

总结

总的来看，vue-router是比较简单的，重点就是路由匹配，编程式导航，钩子函数。这篇只是一个vue-router的实用的知识点的梳理讲解，成文有点杂，哈哈，望见谅哈。