Web Application Development

2020
Vue Router
Vue Router

In a JavaScript web application, a router is the part that syncs the currently displayed view with the browser address bar content.
Vue Router

Features

- Nested route/view mapping
- Modular, component-based router configuration
- Route params, query, wildcards
- Fine-grained navigation control
- Links with automatic active CSS classes
- HTML5 history mode or hash mode, with auto-fallback in IE9
- Customizable Scroll Behavior
Vue Router

Example

```html
<div id="app">
  <h1>Hello App!</h1>
  <p>
    <!-- use router-link component for navigation. -->
    <!-- specify the link by passing the `to` prop. -->
    <!-- `<router-link>` will be rendered as an `<a>` tag by default -->
    <router-link to="/foo">Go to Foo</router-link>
    <router-link to="/bar">Go to Bar</router-link>
  </p>
  <!-- route outlet -->
  <!-- component matched by the route will render here -->
  <router-view></router-view>
</div>
```
// 1. Define some routes
// Each route should map to a component.
const routes = [
    { path: '/foo', component: Foo },
    { path: '/bar', component: Bar }
]
// 2. Create the router instance and pass the `routes` option
const router = new VueRouter({
    routes // short for `routes: routes`
})
// 3. Create and mount the root instance.
// Make sure to inject the router with the router option to make the
// whole app router-aware.
const app = new Vue({
    router
}).$mount('#app')
<table>
<thead>
<tr>
<th>pattern</th>
<th>matched path</th>
<th>$route.params</th>
</tr>
</thead>
<tbody>
<tr>
<td>/user/:username</td>
<td>/user/evan</td>
<td>{ username: 'evan' }</td>
</tr>
<tr>
<td>/user/:username/post/:post_id</td>
<td>/user/evan/post/123</td>
<td>{ username: 'evan', post_id: '123' }</td>
</tr>
</tbody>
</table>
Matching Priority

<table>
<thead>
<tr>
<th>pattern</th>
<th>matched path</th>
<th>$route.params</th>
</tr>
</thead>
<tbody>
<tr>
<td>/user/:username</td>
<td>/user/evan</td>
<td>{ username: 'evan' }</td>
</tr>
<tr>
<td>/user/:username/post/:post_id</td>
<td>/user/evan/post/123</td>
<td>{ username: 'evan', post_id: '123' }</td>
</tr>
<tr>
<td>/user/:username/post/:id</td>
<td>/user/evan/post/123</td>
<td>{ username: 'evan', post_id: '123' }</td>
</tr>
</tbody>
</table>

Sometimes the same URL may be matched by multiple routes. In such a case the matching priority is determined by the order of route definition: the earlier a route is defined, the higher priority it gets.
Nested Routes

/user/foo/profile
+----------------+  /user/foo/posts
| User            |  | User            |
|                 |  |                 |
| Profile         |  | Posts           |
|                 |  |                 |
|                 |  |                 |
Nested Routes

```javascript
routes: [
  {
    path: '/user/:id', component: User,
    children: [
      {
        // UserProfile will be rendered inside User's <router-view>
        // when /user/:id/profile is matched
        path: 'profile',
        component: UserProfile
      },
      {
        // UserPosts will be rendered inside User's <router-view>
        // when /user/:id/posts is matched
        path: 'posts',
        component: UserPosts
      }
    ]
  }
]
```
Programmatic Navigation

```javascript
// literal string path
router.push('home')
// object
router.push({ path: 'home' })
// named route
router.push({ name: 'user', params: { userId: '123' } })
// with query, resulting in /register?plan=private
router.push({ path: 'register', query: { plan: 'private' } })

const userId = '123'
router.push({ name: 'user', params: { userId } }) // -> /user/123
router.push({ path: `/user/${userId}` }) // -> /user/123
// This will NOT work
router.push({ path: '/user', params: { userId } }) // -> /user
```

Note: Inside of a Vue instance, you have access to the router instance as $router. You can therefore call this.$router.push
**Programmatic Navigation**

`router.go(n)`

This method takes a single integer as parameter that indicates by how many steps to go forwards or go backwards in the history stack.

```javascript
// go forward by one record, the same as history.forward()
router.go(1)

// go back by one record, the same as history.back()
router.go(-1)

// go forward by 3 records
router.go(3)

// fails silently if there aren't that many records.
router.go(-100)
router.go(100)
```
Vuex
Vuex is a state management pattern + library for Vue.js applications. It serves as a centralized store for all the components in an application, with rules ensuring that the state can only be mutated in a predictable fashion.
Vuex
Vuex

- Receives new State
- Updates State
const app = new Vue(
{
  data: {
    ... 
    },
    methods: {
    ... 
    },
    computed: {
    ... 
    }
})

const store = new Vuex.Store(
{
  state: {
    ... 
    },
    mutations: {
    ... 
    },
    actions: {
    ... 
    },
    getters: {
    ... 
    }
})
References

https://vuex.vuejs.org/

https://router.vuejs.org/
Questions?

Next: Vue.js