PAU CASESNOVES
Centre Integrat de Formació Professional

Desenvolupament d'aplicacions web

Desenvolupament web a l'entorn client

UT 8.4: Vue - Rutes

Joan Pons Tugores

Índex de continguts

Router

Les aplicacions SPA funcionen capturant els diferents enllaços que porten a les distintes parts de l'aplicació i canviant part del contingut de la pàgina en resposta a aquests enllaços, de manera que l'usuari, i el seu navegador, veu aquests canvis com si hagues navegat a una altra pàgina.

Un router, en una aplicació web, és una llibreria que mapeja les distintes url's als components de l'aplicació.

Vue.js té la llibreria *Vue Router* per fer aquesta tasca.

Instal·lació

Podem instal·lar vue router amb *npm*.

```
npm install vue-router@4
```

Dins la carpeta *src* cream la carpeta *router* i dins ella el fitxer *index.js*. El contingut d'aquest fitxer serà el següent:

```
import { createRouter, createWebHistory } from "vue-router";

// imports de components vista

const routes = [
        {
                path: "ruta que estam especificant",
                name: "nom de la ruta",
                component: "nom del component",
        },
];
```

```
export default createRouter({
        history: createWebHistory(process.env.BASE_URL),
        routes,
});
```

Més endavant veurem com definir les distintes rutes de l'aplicació.

Per afegir el *router* a l'aplicació ho feim al *main.js*:

```
import { createApp } from 'vue'
import { createPinia } from 'pinia'

import App from './App.vue'
import router from './router'

import './assets/main.css'

const app = createApp(App)

app.use(createPinia())
app.use(router)

app.mount('#app')
```

L'importam i l'afegim a l'aplicació.

Estructura de l'aplicació

La idea del router és poder modificar la pàgina proporcionant distints enllaços que, en clicar-los, modifiquen el contingut de la pàgina, al manco de part de la pàgina.

Per aconseguir això necessitam veure una sèrie de conceptes: *view*, *router-link* i *router-view*.

view

Són els components que contenen el disseny de cada una de les parts que formen l'aplicació. És el que vue ens mostrarà en pitjar un determinat link. En realitat són components, però que contenen el que s'ha d'incloure a la pàgina en demanar una determinada ruta.

Com qualsevol altra component contenen html, altres components, javascript, estils, ...

Si volguéssim transformar l'aplicació dels apartats anteriors a una aplicació utilitzant router podríem tenir una vista per l'apartat dels botons com la següent:

```
<template>
  <div>
    <h2>Botons</h2>
    <p>     <ButtonCounter />    </p>
    <p>     <ButtonCounter />    </p>
  </div>
</template>

<script setup>
        import ButtonCounter from "@/components/ButtonCounter";
</script>

<style scoped></style>
```

Podríem crear una vista semblant a aquesta per a cada un dels apartats de l'aplicació.

router-link

En lloc d'utilitzar enllaços HTML utlitzam aquests elements. Això permet a Vue Router gestionar aquests enllaços sense recarregar la pàgina: canviar la url, generar i codificar aquesta url, ..

```
<router-link to="/">Go to Home</router-link>
```

L'atribut *to* conté una de les url's definides al *router*. Vue interpretarà aquest element i farà els canvis necessaris a la pàgina per mostrar el component assignat a la url.

Cada una de les vistes de l'aplicació tendrà al manco una vista amb un *router-link* que l'enllaci.

router-view

És un element de l'html, normalment el posarem dins el *template* de *App.vue*. La seva funció és la de proporcionar l'element dins del qual es mostraran els components que es mostraran en resposta a un link o a un altre. És l'element que conté la part que canvia en pitjar un link.

Quan l'usuari pitgi un enllaç d'una ruta, la vista relacionada amb aquella ruta es mostrarà dins d'aquest element.

Definició de les rutes

Per a cada una de les rutes tendrem un objecte amb tres propietats:

- **path**: La ruta que estam especificant, per exemple */botons*

```
<router-link to="botons"> Botons </router-link>
```

- **name**: El nom de la ruta. Podem utilitzar-lo per especificar la ruta en els *router-link* en lloc del *path*. Té l'avantatge que si canviam la ruta no necessitam actualitzar tots els links.

```
<router-link :to="{name:'botons.show'}"> Botons </router-link>
```

- **component**: El nom del component que s'ha de mostrar en resposta a la ruta.

Un exemple de ruta complet seria:

```
{
  path: "/botons",
  name: "botonsName",
  component: "BotonsSenzills",
},
```

L'exemple anterior defineix una ruta a la que podem accedir des dels elements *router-link* o bé amb el *path* /botons o amb el *name* botonsName.

En resposta a la petició d'aquesta ruta, *vue* montarà el component *BotonsSenzills* dins l'element *router-view*.

Definició del router

Crearem el router dins el fitxer */router/index.js* amb la funció *createRouter*. Aquesta funció rep com a paràmetre un objecte amb les opcions:

```
export default createRouter({
        history: createWebHistory(process.env.BASE_URL),
        routes,
});
```

Les opcions del router són:

- **history**: Com es generen les url's i com afecten a l'histoial del navegador i al SEO
  - createWebHashHistory: La ruta s'afegeix al final de la ruta de l'aplicació amb un #. Evita que el navegador carregui la pàgina en canviar la url. Perjudica el SEO.

    ```
    https://my.app#formulari
    ```

  - createWebHistory: La ruta es canvia totalment com si fos una aplicació multi-pàgina. Funciona amb el SEO.

    ```
    https://my.app/formulari
    ```

    Té un problema: Si l'usuari posa directament una de les URL's la petició arribarà al servidor i com que el servidor no la pot servir donarà un 404.La solució és configurar el servidor de manera que si la url que ha rebut no es correspon amb cap que pugui servir torni *index.html* que és on tenim la nostra aplicació.

- **routes**: La definició de les rutes, la constant que hem creat abans.

Utilització del router

Si programam la nostra aplicació amb *Vue* estam programant la nostra aplicació amb components. Amb Vue router hem d'establir quin component respon a cada petició i on es mostraran aquests components.

Per definir on es mostraran els components hem d'incloure l'element <*router-view*> dins el template de *App.vue*.

Els enllaços es crearan amb l'element *<router-link to="xxx"*> L'HTML de dins d'aquest element serà el que es veurà a l'enllaç. El valor de l'atribut *to* indica la ruta que s'utilitzarà en seguir l'enllaç.

```
<router-link to="/botons"> Botons </router-link>
```

Si volem utilitzar el nom de la ruta en lloc de l'adreça:

```
    <router-link :to="{name:'botonsNom'}">Llista d'aplicacions</router-link>
```

L'element *<router-view/>* marcarà el lloc on es mostraran els components que responen a les rutes.

```
<template>
  <p><img alt="Vue logo" src="./assets/logo.png" /></p>
  <h1>Aplicació de demostració de <em>Vue Router</em></h1>
  <div>
    <router-link to="/">Inici</router-link>
    <router-link to="botons">Botons</router-link>
    <router-link to="propietats">Botons amb propietats</router-link>
    <router-link to="iteracio">Botons dins iteració</router-link>
    <router-link to="getters">Getters</router-link>
  </div>
  <router-view />
```

```
</template>
<script>
export default {
  name: "App",
};
</script>

<style>
#app {
  font-family: Avenir, Helvetica, Arial, sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
  color: #2c3e50;
  margin-top: 60px;
}

a {
  margin-left: 2em;
  margin-right: 2em;
}
</style>
```

Rutes dinàmiques

Moltes vegades necessitam passar informació dins la ruta, per exemple l'id de l'usuari que volem utilitzar al component.

Aquestes parts dinàmiques de la url van precedides per *:*

```
/user/:username
/user/:username/post/:post_id
```

Podem recuperar-les al nostre codi amb

```
<script setup>
        import {useRoute} from "vue-router";
        const route = useRoute()
        route.params
</setup>
```

torna un objecte on cada atribut té el nom que hem posat a la part dinàmica de la ruta i el valor és el que hem rebut dins la url.

pattern	matched path	route.params
/user/:username	/user/evan	{ username: 'evan' }
/user/:username/ post/:post_id	/user/evan/post/123	{ username: 'evan', post_id: '123' }

Quan volem utilitzar una ruta amb paràmetres, és millor utilitzar el name al router-link:

```
<router-link :to="{name:'mostrarParams.show',params:{nom:'Jo'}}">
```

Per exemple, per crear un link per a cada valor d'un array:

```
<router-link
  v-for="n in noms" :key="n"
  :to="{name:'mostrarParams.show',params:{nom:n}}"> {{n}} </router-link>
```

Query string

Si a la ruta afegim dades al query string, podem recuperar-les amb *route.query*. Es tracta d'un objecte on cada un dels elements del query string es converteix en un atribut d'aquest objecte.

ruta	route.query
/user?username=evan	{ username: 'evan' }
/user?username=evan&post_id=123	{ username: 'evan', post_id: '123' }

```
<router-link :to="{name:'mostrarParams.show',query:{nom:'Jo'}}">
```

Per exemple, per crear un link per a cada valor d'un array:

```
<router-link
  v-for="n in noms" :key="n"
  :to="{name:'mostrarParams.show',query:{nom:n}}"> {{n}} </router-link>
```

Navegació per codi

De vegades volem navegar a un altre component com a resposta a un esdeveniment, per exemple en pitjar un botó. Ho farem amb el mètode *push* del router. Per exemple:

- podem utilitzar un path literal

```
<script setup>
import {useRouter} from "vue-router";
const router=useRouter()
</script>
<template>
  //Ruta generada: /comptador
  <p><button @click="router.push('/comptador')"> Ves al
            comptador</button></p>
</template>
```

- podem utilitzar el nom de la ruta i passar dades per query string. Per exemple, si *mostraIlla* és el nom de la ruta i *this.valor='071'*:

```
<script setup>
import {useRouter} from "vue-router";
const router=useRouter()
</script>
<template>
//Ruta generada: mostraIlla?id=071
  <p><button @click="router.push({
            name: 'mostraIlla',
            query: {id:this.valor}
            })"> Ves al comptador</button></p>
</template>
```

- podem utilitzar el nom de la ruta i passar dades al path. Per exemple, si *mostraIlla* és el nom de la ruta i *this.id='071'*:

```
<script setup>
import {useRouter} from "vue-router";
const router=useRouter()
</script>
<template>
//Ruta generada: mostraIlla/071
  <p><button @click="router.push({
            name: 'mostraIlla',
            params: {id:this.valor}
            })"> Ves al comptador</button></p>
</template>
```