

Pages and Routing

Web Development ប្រើប្រាស់ Nuxt 4 ឬ Vue.js, Routing ដើម្បីគ្រប់គ្រងការតភ្ជាប់ URL ទៅកាន់ទំព័រណាមួយ ឬ Component ណាមួយ។ **Nuxt** បង្កើត route ស្វ័យប្រវត្តិ file មាននៅក្នុង `pages/` ។

- នៅពេលអ្នកវាយ `fff-adm.com/about`, Routing នឹងទៅរកមើល File `about.vue` ដើម្បីបង្ហាញឱ្យអ្នកឃើញ។
- នៅពេលអ្នកវាយ `fff-adm.com/contact`, Routing នឹងទៅរកមើល File `contact.vue`។

1. Static Routing

- Index Route: File `app/pages/index.vue` ជាតំណភ្ជាប់ទំព័រដើម (`/`)។
- Static Routes: File `app/pages/about.vue` ជាតំណភ្ជាប់ផ្លូវទៅកាន់ `/about`។
- Nested Routes: បើអ្នកបង្កើត Folder `app/pages/blog/index.vue` វាជាតំណភ្ជាប់ `/blog`។

2. Dynamic Routing - ការប្រើប្រាស់ Parameters ជាមួយតំណភ្ជាប់

ដើម្បីបោះពុម្ពផ្សាយទិន្នន័យ URL ត្រូវមាន Parameters ដូចជា ID ឬ ឈ្មោះ (ឧទាហរណ៍៖ `/posts/1` ឬ `/posts/abc`)។ យើងប្រើសញ្ញា Square Brackets `[]` ដើម្បីបង្កើត dynamic route។

- File: `app/pages/users/[id].vue`
- URL: `/users/123`
- របៀបទាញយកទិន្នន័យពី URL: យើងប្រើ `useRoute()` ។

ឧទាហរណ៍៖

```
-| pages/  
---| users/  
-----| [id].vue
```

```
<script setup>  
const route = useRoute()  
const userId = route.params.id  
</script>  
  
<template>  
<div>  
  <h1>User Profile</h1>  
  <p>User ID របស់អ្នកគឺ: {{ userId }}</p>  
</div>  
</template>
```

3. NuxtLink

ដើម្បីឆ្លងទៅកាន់ទំព័រផ្សេងៗដោយមិនបាច់ Reload Page យើងប្រើ Component `<NuxtLink>` ជំនួសឱ្យ Tag `<a>` ។

```
<template>  
<nav>  
  <NuxtLink to="/">ទៅកាន់ទំព័រដើម</NuxtLink>  
  <NuxtLink to="/about">អំពីយើង</NuxtLink>  
  <NuxtLink :to="/users/${userId}">ទៅកាន់ Profile</NuxtLink>  
</nav>  
</template>
```

4. 404 Page

- File: `app/pages/[...slug].vue`

edited article by Mr. limvichet, contact 012856917/067878118.

- វានឹងបង្ហាញរាល់ពេលដែល User វាយ URL ឆ្គង។

```
<script setup>
const route = useRoute( )
const pathArray = route.params.slug
</script>

<template>
<div class="container">
<h1>ទំព័រ Catch-all</h1>
<p>អ្នកកំពុងស្ថិតនៅផ្លូវ៖ <strong>{{ pathArray.join('/') }}</strong></p>

<div class="debug">
<h3>ទិន្នន័យក្នុង Slug Array:</h3>
<ul>
<li v-for="( folder, index ) in pathArray" :key="index">
ជាន់ទី {{ index + 1 }}: {{ folder }}
</li>
</ul>
</div>

<NuxtLink to="/" class="btn">ត្រឡប់ទៅទំព័រដើម</NuxtLink>
</div>
</template>

<style scoped>
.container { padding: 20px; font-family: sans-serif; }
.debug { background: #f4f4f4; padding: 10px; margin-top: 20px; }
.btn { color: blue; text-decoration: underline; }
</style>
```

5. useRoute

useRoute ត្រូវបានប្រើសម្រាប់ទាញយកព័ត៌មានអំពី URL ដែលកំពុងបង្ហាញនៅលើ Browser បច្ចុប្បន្ន។ អ្នកមិនអាចប្រើវាដើម្បីប្តូរទំព័របានទេ គឺសម្រាប់តែ "មើល" ប៉ុណ្ណោះ។

- params**: តម្លៃប្រែប្រួលក្នុង URL ឧទាហរណ៍៖ id ក្នុង `/user/[id]`
- query**: តម្លៃបន្ទាប់ពីសញ្ញាសួរ ? ឧទាហរណ៍៖ search ក្នុង `?search=nuxt`
- path**: ជា URL
- name**: ឈ្មោះរបស់ Route

```
<script setup>
const route = useRoute( )

// ប្រសិនបើ URL គឺ /blog/my-post?lang=km
console.log( route.params.slug ) // លទ្ធផល: "my-post"
console.log( route.query.lang ) // លទ្ធផល: "km"
</script>

<template>
<p>អ្នកកំពុងមើលអត្ថបទ៖ {{ route.params.slug }}</p>
</template>
```

6. useRouter

useRouter គឺជា tool សម្រាប់បញ្ជាទិញ App របស់អ្នកកំពុងអភិវឌ្ឍ។ វាផ្តល់នូវមុខងារជាច្រើនដូចជា៖ automatic routing, server-side rendering, site generation, and full-stack

```
<script setup>
const router = useRouter( )

const goToContact = ( ) => {
  // ប្តូរទៅទំព័រ contact
  router.push( '/contact' )
}

const goBack = ( ) => {
  // ថយក្រោយ ១ ជំហាន
  router.back( )
}
</script>

<template>
<button @click="goToContact">ទៅកាន់ Contact</button>
<button @click="goBack">ថយក្រោយ</button>
</template>
```

7. navigateTo

ទោះបីជាយើងអាចប្រើ `router.push()` បានក៏ដោយ ប៉ុន្តែក្នុង Nuxt 4 គេណែនាំឱ្យប្រើ Utility មួយឈ្មោះថា `MapsTo()` ជំនួសវិញ ព្រោះវាដំណើរការបានល្អទាំងនៅលើ Server-side និង Client-side។

```
await navigateTo( '/dashboard' )
```

លំហាត់

ចូរអ្នកបង្កើត Project តូចមួយដែលមានរចនាសម្ព័ន្ធ Routing ដូចខាងក្រោម៖

- `app/pages/index.vue` ទំព័រដើម
- `app/pages/books/[id].vue` ទំព័រព័ត៌មានលម្អិតសៀវភៅ
- `app/pages/search.vue` ទំព័រស្វែងរក
- `app/pages/[...slug].vue` ទំព័រសម្រាប់ចាប់ URL ដែលមិនត្រឹមត្រូវ