7/20 ~ 7/24 3주차 - 4

Vue-Router 살펴보기 (2)

vue router로 데이터 전달하기

7/21일 화요일

동적 라우팅 매칭과 beforeEnter, watch등 사용을 해서 데이터를 일정 상세페이지에 전달하려고 하였다.

watch 를 사용해 라우터가 변경 시점을 찾고 나서 다음과 같은 코드로 적용을 했다

```
watch: {
    '$route' (to, from) {
      // router-view에 있는 컴포넌트에서 경로 변경에 반응을 감지 할때 실행된다.
      if(to.name === "CardDetail") {
        console.log("dasdsajkdnjasndsaj")
        this.fetchData();
      }
    }
     methods: {
        fetchData () {
          axios.get("/api/post/" + this.$route.params.id).then((result) => {
            this.cardDetailInfo = result.data
            // console.log(this.cardDetailInfo)
          })
        }
      }
}
```

하지만 axios를 이용해 게시글의 세부 정보까지 가져오는 것보다 라우터로 전환된 일정 상세페이지가 먼저 화면에 나타는 건지 데이터가 넘어가지 않는 건지 알 수가 없었다.....

일정 상세페이지에서 created()에서의 콘솔을 찍어봐도 찍히지 않았기 때문이다.

에러는 일정 상세페이지에서 {{cardDetailInfo}}를 사용할 때 나타났고 <router-view v-bind:cardDetailInfo="cardDetailInfo"을 적용해보고 props로 데이터를 받아봤지만 받지 못하였다.

몇 시간의 검색 끝에 vue router로 데이터 전달하는 방법을 알게 되었다.

vue router로 데이터 전달하기

중요하니 기억해두자.

두 가지 방법이 있다.

vue router로 데이터를 전달하는 방법

1. query
2. params

형태

```
{name: 'Query', query: {name: 'cat', age: 3}

{name: 'Params', query: {name: 'dog', age:4}}
```

```
<template>
  <div>
    main

    <ul>
      <li @click="clickList">
        Query 프로그래밍 방식
      </li>
      <router-link :to="{name: 'Query', query: {name: 'cat', age: 3}}">
        Query 선언적 방식
      </router-link>
      <li @click="clickParams">
        params
```

```
      </li>
      <router-link :to="{name: 'Params', params: {name: 'dog', age:4}}">
        params 선언적 방식
      </router-link>
    </ul>
  </div>
</template>

<script>
    export default {
        name: 'Main',
        methods: {
            clickList () {
                this.$router.push({name: 'Query', query: {name: 'cat', age: 3}})
            },
            clickParams () {
                this.$router.push({name: 'Params', params: {name: 'dog', age:4}})
            }
        }
    }
</script>
```

스크립트를 보면 동적 라우팅 매칭에서는 this.$router.push를 사용하여 변경할 url을 만들어준다

"localhost:3000/detail/:id 이러한 형태이고

```
this.$router.push("/detail/" + this.card.id)
```

이러한 형태로 메소드를 사용하여 url을 바꿔주었다.

하지만 Params 방식으로 데이터를 전달하기위해서.

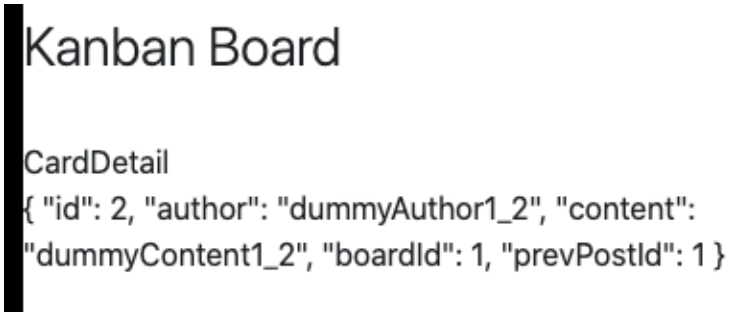
```
this.$router.push({ name: 'CardDetail',params: {cardDetailInfo: this.cardDetailInfo }})
```

이런 식으로 카드 객체(cardDetailInfo가 일정이며 카드이다.)를 담아 보내고 변환된 url은 http://localhost:3000/detail이 된다.

원래는 App.vue(router-view 태그가 있는 곳)에서 axios 요청을 하려고 하였지만 Card.Vue에서 카드 객체에 세부 일정을 담아 일정 상세 페이지에 보내야만 했다.

```
moveToDetailPage: function () {
      axios.get("/api/post/" + this.card.id).then((result) => {
      this.cardDetailInfo = result.data
      this.$router.push({ name: 'CardDetail',params: {cardDetailInfo: this.cardDetailfo }})
 });
```

일정 상세페이지에서 값이 잘 넘어오는 지 {{ $route.params.cardDetailInfo }} 를 사용해서 확인하였다



이제 일정 상세페이지를 만들수 있는 데이터를 일정 상세페이지에서 사용할 수 있게 되었다.

일정내용이 변경이 일어나도 상세페이지를 조회할때마다 새롭게 불러오기 때문에 문제가 없을 것 같다.

그리고 빠진 게 있는 데

index.js 에 있는 routes에

```
const routes = [{
        path: "/",
        component: MainBoard,
        name: "MainBoard", // name을 적어주면 나중에 프로그래밍 방식의 라우터 전환이 편해진다.
    },
    {
        path: "/detail",
        component: CardDetail,
```

```
        name: "CardDetail", // name을 적어주면 나중에 프로그래밍 방식의 라우터 전환이 편해진다.
        props: true // 바로 이 prop를 추가해주면 !!!!!!!!!
    },
```

- 이런식으로 사용이 가능하다. props면 코드가 좀 더 간략히 된다는 걸 기억해 두자.

```
<template>
  <div>
    <h1>Params</h1>

    <h2>params로 받은 데이터</h2>
    <h2>name: {{ $route.params.name }}</h2>
    <h2>age: {{ $route.params.age }}</h2>

    <h2>props로 받은 데이터</h2>
    <h2>name: {{ name }}</h2>
    <h2>age: {{ age }}</h2>
  </div>
</template>
```

내일은 상세페이지에서 내용을 수정하고 디비에 업데이트하고 MainBoard.vue에서 이를 watch 하여 다른 사용자가 변경된 걸 볼 수 있도록 하자.

만약에 이게 빠르게 진행이 된다면 toast editer를 적용하는 것을 얼른 해보자.