(디지털 컨버전스) 스마트 콘텐츠와 웹 융합 응용 SW개발자 양성과정

훈련기간 : 2021.05.07 ~ 2021.12.08

KH정보교육원 | 전승리

게시판 Modify

BoardReadPage.vue

```vue
<template>
    <div align="center">
        <h2>Vue + Spring 게시판 읽기</h2>
        <board-read v-if="board" :board="board"/>
        <p v-else>로딩중 ..... </p>
        <router-link :to="{ name: 'BoardModifyPage', params: { boardNo } }">
            게시물 수정
        </router-link>
        <button @click="onDelete">삭제</button>
        <router-link :to="{ name: 'BoardListPage' }">
            게시물 보기
        </router-link>
    </div>
</template>
```

name: 'BoardModifyPage', params: { boardNo }
path대신 name을 사용하고 params는 전달한 값을 객체로 알려준다.

router

index.js

```js
{
  path: '/board/:boardNo/edit',
  name: 'BoardModifyPage',
  components: {
    default: BoardModifyPage
  },
  props: {
    default: true
  }
```

가져온 boardNo가
/board/:boardNo/edit 이 부분에 매칭된다. (가변데이터 처리)
+ params를 받기위해서는 props와name을 설정해줘야한다.

BoardModifyPage.vue

```vue
<template>
  <div align="center">
      <h2>게시물 수정</h2>
      <board-modify-form v-if="board" :board="board" @submit="onSubmit" />
      <p v-else>로딩중 </p>
  </div>
</template>
```

기존에 readboard에서 가져온 board를 전달

BoardModifyForm.vue

```vue
<form @submit.prevent="onSubmit">
    <table>
        <tr>
            <td>글번호</td>
            <td><input type="text" :value="board.boardNo" disabled></td>
        </tr>
        <tr>
            <td>등록일자</td>
            <td><input type="text" :value="board.regDate" disabled></td>
        </tr>
        <tr>
            <td>제목</td>
            <td><input type="text" v-model="title"></td>
        </tr>
        <tr>
            <td>작성자</td>
            <td><input type="text" :value="board.writer" disabled></td>
        </tr>
        <tr>
            <td>본문</td>
            <td><textarea cols="50" rows="20" v-model="content"></textarea></td>
        </tr>
    </table>
  <div>
  <button type="submit" >수정완료</button>
```

```vue
data() {
    return {
        content: '',
        title: ''
    }
},
methods: {
    onSubmit() {
        const { title, content } = this
        this.$emit('submit', { title, content})
    }
},
created () {
    this.title = this.board.title
    this.content = this.board.content
}
```

미리 title과 content를 가져온 후 v-model로 변경

서버에 put맵핑으로 요청하고 spring에서 update 한다음 다시 board값을 가져온다.

게시판 DELETE

BoardReadPage.vue

```vue
<template>
    <div align="center">
        <h2>Vue + Spring 게시판 읽기</h2>
        <board-read v-if="board" :board="board"/>
        <p v-else>로딩중 ..... </p>
        <router-link :to="{ name: 'BoardModifyPage', params: { boardNo } }">
            게시물 수정
        </router-link>
        <button @click="onDelete">삭제</button>
        <router-link :to="{ name: 'BoardListPage' }">
            게시물 보기
        </router-link>
    </div>
</template>
```

```vue
onDelete () {
    const { boardNo } = this.board
    axios.delete(`http://localhost:7777/vueboard/${boardNo}`)
    .then (() => {
        alert('삭제성공')
        this.$router.push({ name: 'BoardListPage'})
    })
    .catch(err => {
        alert(err.response.message)
    })
}
```

```vue
{
  path: '/board',
  name: 'BoardListPage',
  components: {
    default: BoardListPage
  }
```

```vue
@DeleteMapping("/{boardNo}")
public ResponseEntity<Void> delete(@PathVariable("boardNo") Integer boardNo) throws Exception {
    service.remove(boardNo);

    return new ResponseEntity<Void>(HttpStatus.OK);
}
```

삭제하고 이동하고 끝난다. 반환값 없다.
반환값이 없으면 .then(() => 이런식으로 표시해준다.