

(디지털 컨버전스) 스마트 콘텐츠와 웹 융합 응용 SW개발자 양성과정

훈련기간 : 2021.05.07 ~ 2021.12.08

Spring 전송 데이터

```
@PostMapping("/register")
public ResponseEntity<Board>register(@Validated @RequestBody Board board) throws Exception {
    log.info("post register request from vue");

    service.register(board);
    return new ResponseEntity<>(board, HttpStatus.OK);
}
```

JSON.stringify(res))

Status Code: 200

localhost:8080 내용:

등록 성공! - {"data":
{"memberNo":0,"id":"Test","pw":"1234Test","regDate":null},"stat
us":200,"statusText":"","headers":{"content-type":"application/
json"},"config":{"url":"http://localhost:7777/vuemember/
register","method":"post","data":{"id":"Test","pw":
"1234Test"},"headers":{"Accept":"application/json, text/plain, */
*"},"Content-Type":"application/
json;charset=utf-8"},"transformRequest":[null],"transformResponse":
[null],"timeout":0,"xsrCookieName":"XSRF-
TOKEN","xsrHeaderName":"X-XSRF-

확인

board, HttpStatus.OK와 뒤에는 url등이 있는 형식인것같다.

Spring 전송 데이터

List형식으로 반환

```
@GetMapping("/lists")
public ResponseEntity<List<Board>> getLists() throws Exception {
    log.info("getList()" + service.list());

    return new ResponseEntity<List<Board>>(service.list(), HttpStatus.OK);
}
```

alert(JSON.stringify(res))

localhost:8080 내용:

```
{
  "data": [
    {
      "boardNo": 4,
      "title": "제목을 작",
      "content": "본문을 작",
      "writer": "L",
      "regDate": "2021-07-16T10:47:09.000+00:00"
    },
    {
      "boardNo": 3,
      "title": "1글",
      "content": "22",
      "writer": "11",
      "regDate": "2021-07-16T10:36:58.000+00:00"
    },
    {
      "boardNo": 2,
      "title": "제목을 작성",
      "content": "본문을 작성",
      "writer": "",
      "regDate": "2021-07-15T14:46:31.000+00:00"
    },
    {
      "boardNo": 1,
      "title": "spring",
      "content": "select * from",
      "writer": "vue",
      "regDate": "2021-07-15T11:43:09.000+00:00"
    }
  ]
}
```

확인

구글 개발자 모드

```
▼ Watch
▼ res: Object
  ▶ config: {url: "http://localhost:7777/vueboard/lists", method: "get", headers: {...}, transformRequest: Array(1), t...}
  ▼ data: Array(4)
    ▶ 0: {boardNo: 4, title: "제목을 작", content: "본문을 작", writer: "L", regDate: "2021-07-16T10:47:09.000+00:00"}
    ▶ 1: {boardNo: 3, title: "1글", content: "22", writer: "11", regDate: "2021-07-16T10:36:58.000+00:00"}
    ▶ 2: {boardNo: 2, title: "제목을 작성", content: "본문을 작성", writer: "", regDate: "2021-07-15T14:46:31.000+00:00"}
    ▶ 3: {boardNo: 1, title: "spring", content: "select * from", writer: "vue", regDate: "2021-07-15T11:43:09.000+00:00"}
    length: 4
    __proto__: Array(0)
  ▶ headers: {content-type: "application/json"}
  ▶ request: XMLHttpRequest {readyState: 4, timeout: 0, withCredentials: false, upload: XMLHttpRequestUpload, onread...}
  status: 200
  statusText: ""
  __proto__: Object
```

www.localhost:7777/vueboard/lists

```
[
  {
    "boardNo": 4,
    "title": "제목을 작",
    "content": "본문을 작",
    "writer": "L",
    "regDate": "2021-07-16T10:47:09.000+00:00"
  },
  {
    "boardNo": 3,
    "title": "1글",
    "content": "22",
    "writer": "11",
    "regDate": "2021-07-16T10:36:58.000+00:00"
  },
  {
    "boardNo": 2,
    "title": "제목을 작성",
    "content": "본문을 작성",
    "writer": "",
    "regDate": "2021-07-15T14:46:31.000+00:00"
  },
  {
    "boardNo": 1,
    "title": "spring",
    "content": "select * from",
    "writer": "vue",
    "regDate": "2021-07-15T11:43:09.000+00:00"
  }
]
```

Spring 전송 데이터 (가변데이터)

```
@GetMapping("/{boardNo}")
public ResponseEntity<Board> read(@PathVariable("boardNo") Integer boardNo) throws Exception {
    Board board = service.read(boardNo);
    log.info("get board Read" + boardNo);

    return new ResponseEntity<Board>(board, HttpStatus.OK);
}
```

구글 개발자 모드

```
▼ res: Object
  ▶ config: {url: "http://localhost:7777/vueboard/4", method: "get", headers: {...}, tran...
  ▼ data:
    boardNo: 4
    content: "본문을 작성해보시면 됩니다."
    regDate: "2021-07-16"
    title: "제목을 작성해보시면 됩니다."
    writer: "관리자"
    ▶ __proto__: Object
  ▶ headers: {content-type: "application/json"}
  ▶ request: XMLHttpRequest {readyState: 4, timeout: 0, withCredentials: false, upload:...
    status: 200
    statusText: ""
    ▶ __proto__: Object
```

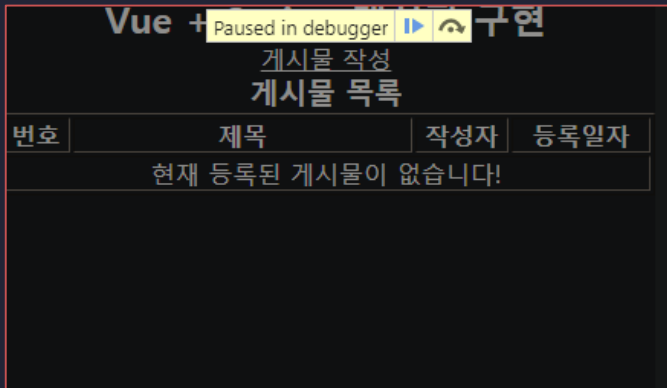
localhost:8080/board/4

```
{
  "boardNo": 4,
  "title": "제목을 작성해보시면 됩니다.",
  "content": "본문을 작성해보시면 됩니다.",
  "writer": "관리자",
  "regDate": "2021-07-16"
}
```

Vue mounted Vs Computed

흐름을 보자면 list페이지를 보여주기위해서 미리 computed가 임계영역에 boards의 배열을 계산해놓는다.
크롬에서 디버깅했는데 computed가 먼저 실행되는거 같은 이유가 맨처음 빈화면으로 로딩된다.
computed에서 맨처음 state에 있는 boards를 아직 요청받기전에 빈 board를 가져와서 그런것같다.
다음과 같은 흐름으로 실행 됐다.

1. computed => 빈 list



번호	제목	작성자	등록일자
현재 등록된 게시물이 없습니다!			

2. mounted => 다시 computed



번호	제목	작성자	등록일자
4	제목을 작성하세요.	LORE	2021-07-16T10:47:09.000+00:00
3	1글	11	2021-07-16T10:36:58.000+00:00
2	제목을 작성하세요.		2021-07-15T14:46:31.000+00:00
1	spring	vue	2021-07-15T11:43:09.000+00:00

그후에 HTML요소를 붙이는 mounted가 실행되는데 이 mounted 안에는 mapAction에 있는 fetchBoardList가 있다.
서버에 요청해서 get한 데이터를 동기화(mutaion)한후에 임계영역 boards 데이터에 넣고 가져오는 순서같다.

Vue mounted Vs Computed

```
<template>
  <div id="board">
    <h2>Vue + Spring 게시판 구현</h2>
    <router-link :to="{ name: 'BoardRegisterPage' }">
      게시물 작성
    </router-link>
    <board-list :boards="boards"/>
  </div>
</template>

<script>
import BoardList from '@components/board/BoardList.vue'
import { mapState, mapActions } from 'vuex'
export default {
  name: 'BoardListPage',
  components: {
    BoardList
  },
  computed: {
    ...mapState(['boards'])
  },
  mounted () {
    this.fetchBoardList()
  },
  methods: {
    ...mapActions(['fetchBoardList'])
  }
}
</script>
```

JS states.js

```
export default {
  // TODO
  todoItems: [],
  editingId: 0,
  nextTodoId: 1,
  filter: null,
  // 몬스터
  monsterElements: [],
  nextMonsterId: 1,
  randomFromSpring: 0,

  //
  boards: [],
  board: null
}
```

6

JS mutations.js

```
[FETCH_BOARD_LIST] (state, boards) {
  state.boards = boards;
},
```

5

JS actions.js

```
fetchBoardList ({ commit }) {
  return axios.get('http://localhost:7777/vueboard/lists')
    .then((res) => {
      commit(FETCH_BOARD_LIST, res.data)
    })
},
```

4

1

7

2

3

Vue mounted Vs Computed

board를 만들때 3가지중 하나에서 실행되도 문제는 없었다.

BeforeMount

```
beforeMount () {  
  this.fetchBoard(this.boardNo)  
    .catch(err => {  
      alert(err.response.data.message)  
      this.$router.push()  
    })  
},
```

mounted

```
mounted () {  
  this.fetchBoard(this.boardNo)  
    .catch(err => {  
      alert(err.response.data.message)  
      this.$router.push()  
    })  
},
```

created

```
created () {  
  this.fetchBoard(this.boardNo)  
    .catch(err => {  
      alert(err.response.data.message)  
      this.$router.push()  
    })  
},
```

VDOM에서 이 세가지는 된다.

3개 전부 fetchboard가 정상적으로 보여진다.
이 차이는 아래 적힌대로인 것 같다.

```
beforeCreate() {  
  console.log('Vue 객체를 만들기 이전입니다.')},  
created() {  
  console.log('Vue 객체를 만들었습니다.')},  
beforeMount() {  
  console.log('HTML 요소를 붙이기 전입니다.')},  
mounted() {  
  console.log('HTML 요소를 붙입니다.')},  
beforeUpdate() {  
  console.log('VDOM의 변화를 감지합니다.')  
  var i  
  for (i = 0; i < this.monsters.length; i++) {  
    if (this.monsters[i].hp <= 0) {  
      this.monsters.splice(i, 1)  
    }  
  }  
},  
updated() {  
  console.log('VDOM의 변화를 적용합니다.')},  
beforeDestroy() {  
  console.log('Vue 객체를 파괴하기 이전입니다.')},  
destroyed() {  
  console.log('Vue 객체를 파괴하였습니다.')}
```

BoardList에서 특정 보드번호를 가변데이터로 맵핑할때

router-link로 BoardReadPage로 이동하면서
params: { boardNo: board.boardNo.toString() } 를 통해 보드 번호가 보내진다. /boardNo

BoardList.vue

```
<template>
  <div>
    <h3>게시물 목록</h3>
    <table border="1">
      <tr>
        <th align="center" width="100">번호</th>
        <th align="center" width="640">제목</th>
        <th align="center" width="150">작성자</th>
        <th align="center" width="240">등록일자</th>
      </tr>
      <tr v-if="!boards || (Array.isArray(boards) && boards.length === 0)">
        <td colspan="4">
          현재 등록된 게시물이 없습니다!
        </td>
      </tr>
      <tr v-else v-for="board in boards" :key="board.boardNo">
        <td align="center">{{ board.boardNo }}</td>
        <td align="left">
          <router-link :to="{ name: 'BoardReadPage',
            params: { boardNo: board.boardNo.toString() },
            {{ board.title }}">
          </router-link>
        </td>
        <td align="right">{{ board.writer }}</td>
        <td align="center">{{ board.regDate }}</td>
      </tr>
    </table>
  </div>
</template>
```

```
<script>
export default {
  name: 'BoardList',
  props: {
    boards: {
      type: Array
    }
  }
}
</script>
```

router

JS index.js

```
{
  path: '/board/:boardNo',
  name: 'BoardReadPage',
  components: {
    default: BoardReadPage
  },
  props: {
    default: true
  }
}
```

path: 'board/ :boardNo' 에서 :boardNo은 가변인자를 뜻한다.
방금 받아온 BoardList.vue에서 받아온 /boardNo로 이동

BoardReadPage.vue

```
created () {
  this.fetchBoard(this.boardNo)
    .catch(err => {
      alert(err.response.data.message)
      this.$router.push()
    })
},
```

받아온 (boardNo을 before, created, beforeMounted)를 통해 실행

catch문 this.\$router.push():
에러 발생하면 메시지이후에 특정 url로 보내겠다는 뜻
오류가 발생할 일 이 없어서 따로 로케이션이 없는것같다.
빠져도 실행이 가능한 하다.

BoardReadPage.vue

```
methods: {
  ...mapActions(['fetchBoard']),
  onDelete () {
  }
}
```

JS actions.js

```
fetchBoard ({ commit }, boardNo) {
  return axios.get(`http://localhost:7777/vueboard/${boardNo}`)
    .then((res) => {
      commit(FETCH_BOARD, res.data)
    })
}
```

get 요청

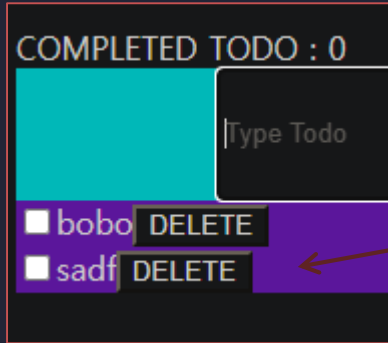
JS mutations.js

```
[FETCH_BOARD] (state, board) {
  state.board = board;
}
```

JS states.js

```
board: null
```


Actions 비동기 처리



```
actions: {  
  addTodo({commit}, text) {  
    setTimeout(function() {  
      commit('ADD_TODO', text)  
    }, 2000);  
    console.log(text)  
  }  
}
```

위와같이 2초간의 딜레이를 두면 console.log의 text가 먼저 실행되고 2초뒤에 addTodo가 된다.
실행은 setTime이 먼저 되지만 기다리지않고 바로 text가 실행됨

