(디지털 컨버전스) 스마트 콘텐츠와 웹 융합 응용 SW개발자 양성과정

훈련기간 : 2021.05.07 ~ 2021.12.08

KH정보교육원 | 전승리

Vue

BoardRegisterPage.vue

```vue
<template>
    <div align="center">
        <h2>게시물 작성</h2>
        <board-register-form @submit="onSubmit"/>
    </div>
</template>

<script>
import BoardRegisterForm from '@/components/board/BoardRegisterForm.vue'
import axios from 'axios'
export default {
    name: 'BoardRegisterPage',
    components: {
        BoardRegisterForm
    },
    methods: {
        onSubmit (payload) {
            const { title, content, writer } = payload
            axios.post('http://localhost:7777/vueboard/register', { title, writer, content })
                .then(res => {
                    alert('등록 성공! - ' + res)

                })
                .catch(res => {
                    alert(res.response.data.message)
                })
        }
    }
}
</script>
```

json 방식이라 post다 만약에 get으로 하면 오류가 발생함.

7777인 스프링서버에 전송

router 설정

list부분은 나중에 구현

index.js

```js
{
  path: '/board',
  name: 'BoardListPage',
  components: {
    default: BoardListPage
  }
},
{
  path: '/board/create',
  name: 'BoardRegisterPage',
  components: {
    default: BoardRegisterPage
  }
}
```

BoardRegisterForm.vue

```vue
<template>
    <form @submit.prevent="onSubmit">
        <h3>게시물 작성 형태</h3>
        <table>
            <tr>
                <td>제목</td>
                <td><input type="text" v-model="title"></td>
            </tr>
            <tr>
                <td>작성자</td>
                <td><input type="text" v-model="writer"></td>
            </tr>
            <tr>
                <td>본문</td>
                <td><textarea cols="50" rows="20" v-model="content"></textarea></td>
            </tr>
        </table>

        <div>
            <button type="submit">등록</button>
            <router-link :to="{ name: 'BoardListPage' }">
                취소
            </router-link>
        </div>
    </form>
</template>

<script>
export default {
    name: 'BoardRegisterForm',
    data () {
        return {
            title: '제목을 작성하세요.',
            writer: '',
            content: '본문을 작성하면 됩니다.'
        }
    },
    methods: {
        onSubmit () {
            const { title, writer, content } = this
            this.$emit('submit', { title, writer, content })
        }
    }
}
</script>
```

prevent : 새로고침 방지

this data 값 전체를 가르킨다.

작성하고 emit으로 전달

Spring

```java
@Slf4j
@Controller
@RequestMapping("/vueboard")
@CrossOrigin(origins = "http://localhost:8081", allowedHeaders = "*")
public class VueBoardController {

    @Autowired
    private VueBoardService service;

    @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);
    }
}
```

ResponseBody Vs RequestBody

RequestBody: HTTP 요청의 body 내용을 자바 객체로 매핑하는 역할을 한다.

ResponseBody: 자바 객체를 HTTP 요청의 body 내용으로 매핑하는 역할은 한다.

@validated : 하나의 클래스 내 모든 객체에 대해 검증한다. (복잡해서 차후에)

entity는 이전 board구조와 같아서 그대로 사용했고 나머지 repository, service 부분도 재활용했지만 테이블을 새로 생성했기때문에 repository에서 날리는 query문은 새로 생성한 테이블에 맞게 수정했다. 같은 entity여도 아마 클래스로만 구현했으면 복잡한 의존성 때문에 수정하기 힘들었을 것이다. 추상메서드만 있고 구현부가 없는 인터페이스의 특징 덕에 몇분이면 수정이가능했다.