(디지털 컨버전스) 스마트 콘텐츠와 웹 융합 응용 SW개발자 양성과정

훈련기간 : 2021.05.07 ~ 2021.12.08

KH정보교육원 | 전승리

props 사용한 예) editingId

TodoList.vue

```vue
<template>
    <div>
        <ul>
            <h3>해야할 일 리스트</h3>
            <todo-item v-for="todoItem in todoItems"
                       v-bind:key="todoItem.id"
                       v-bind:todoItem="todoItem"
                       v-bind:editingId="editingId"
                       v-on:removeTodo="onRemoveTodo"
                       v-on:editTodo="onEditTodo"
                       v-on:setEditingId="SET_EDITTING_ID"
                       v-on:resetEditingId="RESET_EDITTING_ID"
                       v-on:toggleTodoStatus="onToggleTodoStatus"/>
        </ul>
    </div>
</template>
```

TodoItem.vue

```vue
<script>
export default {
    name: 'TodoItem',
    props: {
        // todo에서 v-bind를 통해서 값을 받아옴
        todoItem: {
            type: Object
        },
        editingId: {
            type: Number
        }
    },
```

TodoList.vue

```vue
import { mapState, mapMutations, mapGetters } from 'vuex'
```

TodoList.vue

```vue
computed: {
    ...mapGetters ([
        'filteredTodoItems'
    ]),
    ...mapState ([
        'editingId'
    ]),
    todoItems () {
        return this.filteredTodoItems
    }
},
```

states.js

```vue
export default {
    todoItems: [],
    editingId: 0,
    nextTodoId: 1,
    filter: null,
    monsterElements: [],
    nextMonsterId: 1,
    randomFromSpring: 0
}
```

TodoItem.vue

```vue
computed: {
    isEditing () {
        return this.todoItem.id === this.editingId
    }
},
```

TodoItem.vue

```vue
<span v-if="!isEditing" @dblclick="handleDoubleClick">
```

props : 부모 컴포넌트에서 자식컴포넌트 방향

v-bind:X = "Y" --> 태그로 설정한 컴포넌트에서 X에 Y값을 바인딩한다.

자식 컴포넌트에서 props로 선언한 변수 X에 값의 타입에 맞게 부모 컴포넌트 에서 가져온 Y를 바인딩 한다.

$emit 사용한 예) onEditTodo

$emit : 자식 컴포넌트에서 부모 컴포넌트방향

v-on:X ="Y" --> 태그로 설정한 컴포넌트에서 X의 값을 받아오고 그 값을 Y메서드의 매개변수로 받게된다.

자식 컴포넌트 $emit("X", value)

부모 컴포넌트 method: {Y(value){}}

Todo.vue

```
<template>
    <div class="todo">
        <todo-header></todo-header>
        <todo-input v-on:addTodo="onAddTodo"></todo-input>
        <todo-list
                v-on:removeTodo="onRemoveTodo"
                v-on:editTodo="onEditTodo"
                v-on:toggleTodoStatus="onToggleTodoStatus">
        </todo-list>
        <todo-footer v-on:removeAll="onClearAll"></todo-footer>
    </div>
</template>
```

TodoList.vue

```
v-on:editTodo="onEditTodo"
onEditTodo (content, id) {
    this.$emit('editTodo', content, id)
},
```

TodoItem.vue

```
editTodo (event) {
    const id = this.todoItem.id
    const content = event.target.value.trim()
    if (content.length <= 0) {
        return false
    }
    this.$emit('editTodo', content, id)
    this.$refs.content.blur()
},
```

Todo.vue

```
onEditTodo (content, id) {
    this.editTodo({ id, content })
    this.save()
},
```

states.js

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

```
methods: {
    ...mapActions ([
        'addTodo',
        'removeTodo',
        'editTodo',
        'save',
        'clearAll',
        'toggleTodoStatus'
    ]),
```

mutation-types.js

```
export const EDIT_TODO = 'EDIT_TODO'
```

actions.js

```
editTodo ({ commit }, payload) {
    commit(EDIT_TODO, payload)
},
```

mutations.js

```
[EDIT_TODO] (state, payload) {
    const { id, content } = payload
    const targetIndex = state.todoItems.findIndex(v => v.id === id)
    const targetTodoItem = state.todoItems[targetIndex]
    state.todoItems.splice(targetIndex, 1, { ...targetTodoItem, content })
},
```

{...targetTodoItem, content } 가 무슨뜻인지 기억이 안납니다.

Todo.vue

```
import { mapActions } from 'vuex'
```

질문 : 제일 하위 컴포넌트인 TodoItem.vue에서 action으로 안 넘기고 Todo.vue까지 emit한 이유가 궁금합니다.

Spring - Vue 연결 npm install axios 필수

Test.vue

$store 뷰의 내장 store의 getter에서 randomFromSpring데이터를 가져옴

```
<b> random: {{ this.$store.getters.randomFromSpring }} </b><br>
<input type="button" @click="randomNumber()" value="random"/><br>
```

Test.vue

```
methods: {
    ...mapActions([
        'generateRandomNumber'
    ]),
    randomNumber() {
        this.generateRandomNumber()
    }
}
```

actions.js

```
generateRandomNumber({ commit }) {
    console.log( commit )
    axios.get('http://localhost:7777/random')
    .then((res) => {
        commit(SUCCESS_GEN_RAND_NUM, parseInt(res.data.randNumber))
    })
    .catch((res) => {
        commit(FAIL_GEN_RAND_NUM, res)
    })
},
```

axios.get: GET 요청
axios.post: POST 요청
특정 URL로 GET 혹은 POST, 그외의 요청을 보낼 수 있음
보내고 넘겨 받은 데이터는 , then((res)) 절로 수신함
.catch((res)) 절은 오류가 발생했을경우
어찌 되었든 응답받는 데이터는 res가 가지고 있음

mutations.js

```
[SUCCESS_GEN_RAND_NUM] (state, payload) {
    console.log('payload =' + payload)
    state.randomFromSpring = payload
},
[FAIL_GEN_RAND_NUM] () {
    console.log("통신에러!")
}
```

mutation-types.js

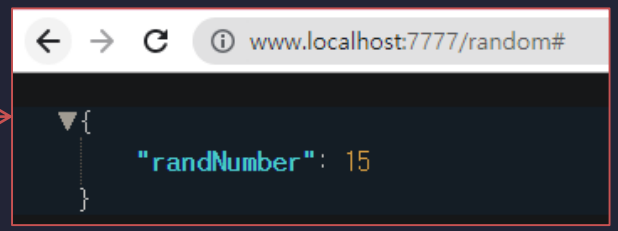
```
export const SUCCESS_GEN_RAND_NUM = 'SUCCESS_GEN_RAND_NUM'
export const FAIL_GEN_RAND_NUM = 'FAIL_GEN_RAND_NUM'
```

Spring - Vue 연결

Vue 포트번호

```java
@Slf4j
@Controller
@CrossOrigin(origins = "http://localhost:8080", allowedHeaders = "*")
public class VueCommContorller {

    @Autowired
    private RandNumService randNumService;

    @GetMapping("/random")
    @ResponseBody
    public ResponseEntity<RandNumMessage> getRandom() {
        log.info("getRandom() request from vue");

        RandNumMessage random = randNumService.getRandom();

        return ResponseEntity.ok(random);
    }
}
```

서버의 정의는 ? 클라이언트에게 서비스제공
클라이언트는 어떤 서비스를 받는가 ? 틀라이언트(사용자)가 요청한 서비스르 받음
사용자가 요청할 URL을 맵핑함(서비스 요청)
ResponseBody가 붙어 있으면 리턴하는 값을 json 형식으로 만들어줌



@CrossOrigin 어노테이션을 붙여주면 기본적으로 '모든 도메인, 모든 요청방식' 에 대해 허용 한다는 뜻이다.
특정 method에만 붙여서 특정 method만 허용할수있다.