Vuex

백성애

https://vuex.vuejs.org/

Vuex란?

- vue.js의 상태관리 패턴 플러그인
- 애플리케이션의 모든 컴포넌트에 대한
- 글로벌 저장소 역할을 하면서
- 사이드 이펙트(side effect) 없이 상태를 변경할 수
- 있다.

- **Side effect란?**
  (자바스크립트) 코드가 외부 세계에 영향을 주거나 받는 것.
  함수가 일관된 결과를 보장하지 못하거나, 함수 외부 어디든 조금이라도 영향을 주는 경우를 의미.

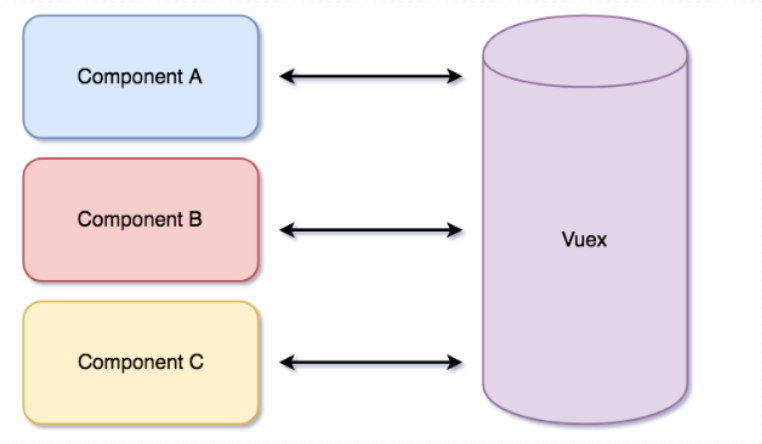
Vuex 란?

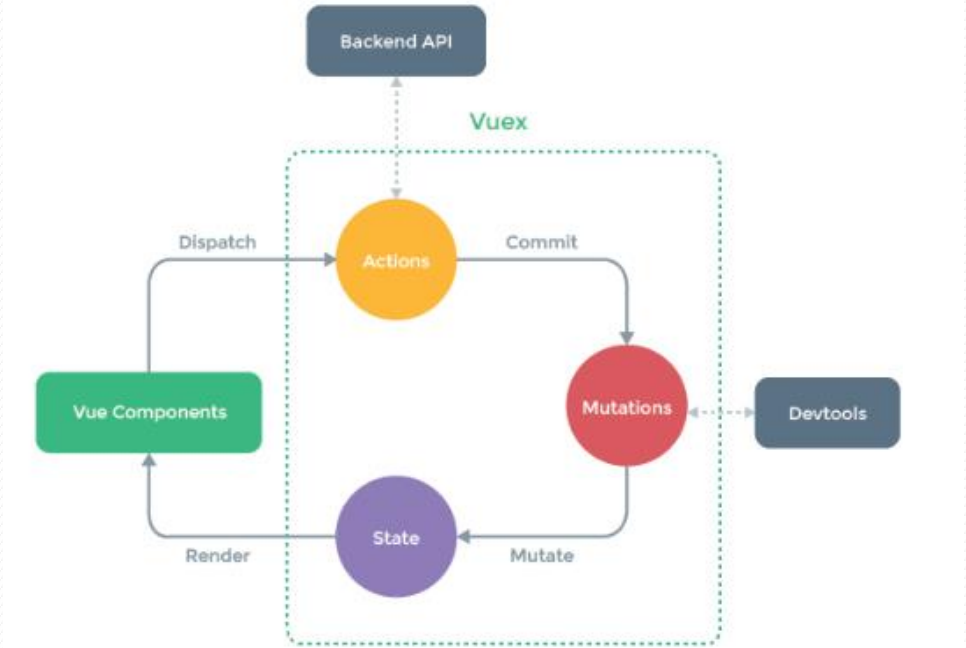
- - Vuex를 사용하는 경우
-
- (1) 여러 컴포넌트가 하나의 데이터를 공유할 때 사용
- (2) 부모 자식 컴포넌트 간의 데이터 공유가 아닌
- 수평적인 컴포넌트간의 데이터 공유가 필요할 때
-
- 이벤트 버스로 해결할 수도 있으나, 애플리케이션의 규모가 커질 때는 Vue 객체를 이용하는 것 보다
- Vuex를 통해 관리하는 것이 효율적이다.

Vuex의 주요 속성

- (1) state : Vuex에는 state라는 하나의 데이터 저장소를 가지고 있다. 어플리케이션의 모든 상태는 하나의 Object 형태로 관리가 된다.
- (2) getters : computed와 비슷
- (3) mutations : state 자체를 변경하는 메소드.
  state를 **동기적으로** 수정할 때 사용.
  mutation을 실행할 때는 commit을 이용
- (4) actions : 사용자의 입력에 따라 데이터를 변경하는 methods.
  Mutation과 비슷하나 **비동기적**으로 수행할 때, 또는 **여러 뮤테이션을 연달아 실행**할 때 사용함

Vuex의 데이터 흐름


Component A
Component B
Component C
Vuex


Backend API
Vuex
Dispatch
Actions
Commit
Vue Components
Mutations
Devtools
State
Render
Mutate

Vuex 설치 및 활용

- Vuex 설치
- npm i vuex --s

- main.js 에서 ➔

```
import Vue from 'vue'
import App from './App2.vue'
import router from './router'
import 'bootstrap';
import 'bootstrap/dist/css/bootstrap.min.css';
import Vuex from 'vuex'

Vue.use(Vuex); //vuex 플러그인 설정

const store =new Vuex.Store({
  state:{
    count:0
  },
  mutations:{
    INCREMENT(state){
      state.count+=1;
    },
    DECREMENT(state){
      state.count-=1;
    }
  }
})

Vue.config.productionTip = false
new Vue({
  store,
  render: h => h(App),
  router:router
}).$mount('#app')
```

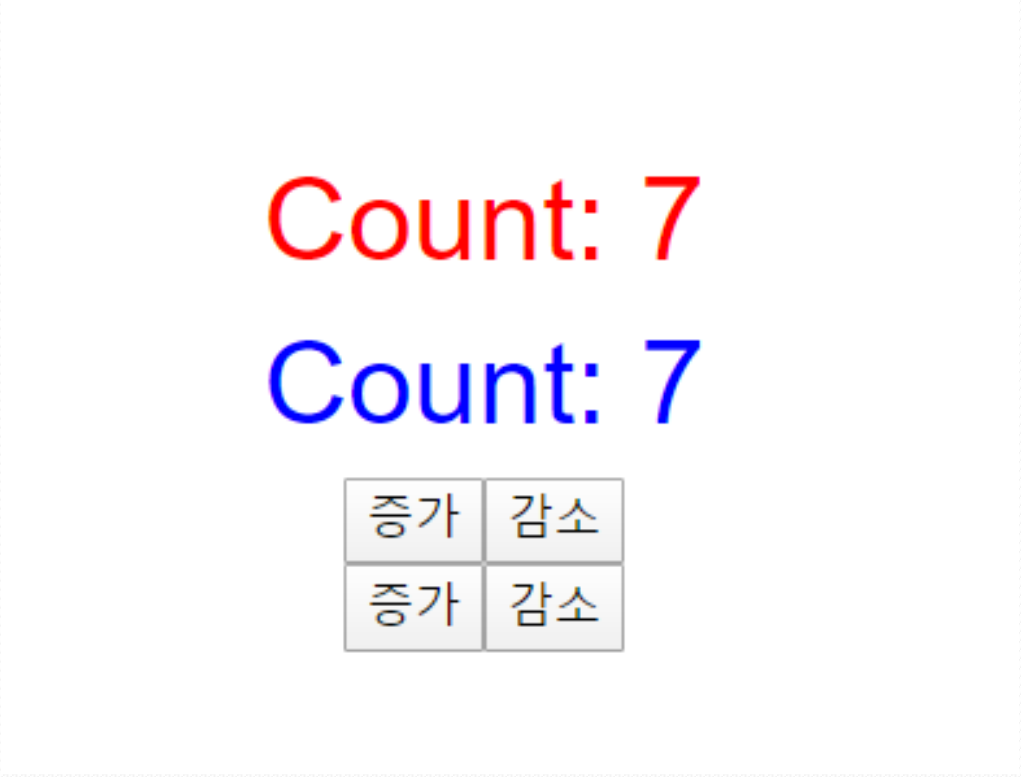
components/Counter.vue

App2.vue

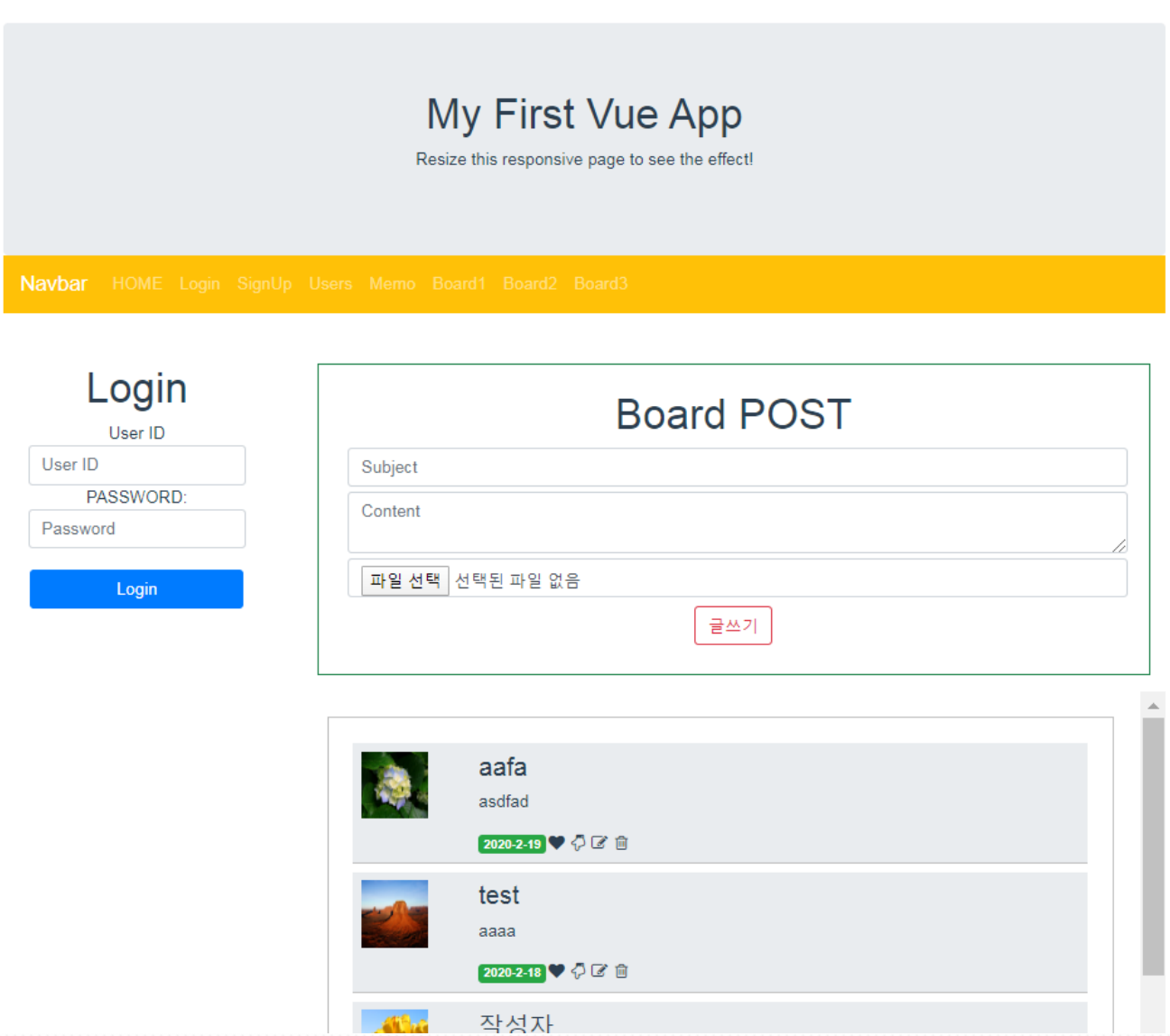
myvue-app > src > App2.vue > {} "App2.vue" > script

```vue
<template>
  <div id="app">
      <h1 style="color:red">Count: {{$store.state.count}} </h1>
      <h1 style="color:blue">Count: {{mycount}} </h1>
    <counter-comp></counter-comp>
    <counter-comp></counter-comp>
  </div>
</template>
<script>
import Counter from "./components/Counter.vue";
export default {
  name: "app",
  components: {
    "counter-comp": Counter,
  },
  computed:{
      mycount(){
          return this.$store.state.count;
      }
  }
  /*COMPUTED를 사용하여 VUEX 의 STATE를 가져올 수 있다.
반드시 computed를 이용해서 state를 가져온다.
컴포넌트들은 Vuex에서 computed를 통해 데이터를 가져와서,
화면에 표시하고
뮤테이션들은 store.commit으로 실행하여 데이터를 수정 한다.
 */
};
</script>
```

브라우저 실행결과



Vuex를 이용한 Board



Vuex를 이용한 Board

- src/store/index.js 를 작성하여 아래와 같이 구성한다.

```
===src/store/index.js============================================
import Vue from 'vue'
import Vuex from 'vuex'
import axios from 'axios'
Vue.use(Vuex);
//vuex 플러그인 설정 Vue와 Vuex를 연결하고,
//또한 main.js에서 Vue인스턴스에 store를 연결한다.

const store =new Vuex.Store({
  state:{ //store가 가지고 있는 상태값으로 Vue인스턴스의 data와 유사함
     mode:'write',
     board:{
         idx:1,
         name:'',
         subject:'',
         content:'',
         filename:'',
     },
     result:false,
     errMsg:'',
     boards:[]
   },
  mutations:{//Vue인스턴스의 메소드와 유사=>상태를 변경하는 내용을 기술
     BOARD_WRITE_SUCCESS(state, result){
       state.result=result;
     },
     BOARD_WRITE_FAIL(state, msg){
       state.errMsg=msg;
     },
   },
  actions:{
    // ['Board_WRITE_SUCCESS']:(store){}
    doWrite({commit}, board){
     ...중략
     let url='http://localhost:9090/VueBackend/boardEnd.do'
```

src/store/index.js

```
22    },
23    mutations:{//Vue인스턴스의 메소드와 유사=>상태를 변경하는 내용을 기술
24      BOARD_WRITE_SUCCESS(state, result){
25        state.result=result;
26      },
27      BOARD_WRITE_FAIL(state, msg){
28        state.errMsg=msg;
29      },
30    },
31    actions:{
32      // ['Board_WRITE_SUCCESS']:(store){}
33      doWrite({commit}, board){
34        ...중략
35        let url='http://localhost:9090/VueBackend/boardEnd.do'
36        axios.get(url).then(function(){
37            alert('success'+this);
38            commit('BOARD_WRITE_SUCCESS',true)
39        })
40        .catch(function(e){
41            alert('fail'+e);
42            commit('BOARD_WRITE_FAIL',e.message)
43        });
44       }//doWrite()----------------
45       ,
46    }//actions
47  })//
48
49  export default store;
50  ====================================================================
```

Src/main.js

```
===main.js============================================================
import Vue from 'vue'
import App from './App.vue'
import router from './router'
import 'bootstrap';
import 'bootstrap/dist/css/bootstrap.min.css';
import store from './store/index.js'


Vue.config.productionTip = false
new Vue({
  store,
  render: h => h(App),
  router:router
}).$mount('#app')
=======================================================================
```

Store/index.js를 import하여 store로 받은 뒤 이를 Vue객체에 넣어준다.

BoardWrite.vue작성

```
======BoardWrite.vue================================================================
<template>
    <div class="container mt-5 mb-3">
        <div class="row">
            <div class="col-md-12 p-4" style="border:1px solid seagreen">
                <h1>Board POST</h1>
                <form id="bf" class="form" method="post" enctype="multipart/form-data" v-on:submit="handleSubmit">

                    <input type="text" name="subject" id="subject" v-model="board.subject"
                     class="form-control m-1" placeholder="Subject">

                    <textarea name="content" placeholder="Content"
                    v-model="board.content"  class="form-control m-1"></textarea>
                    <input type="file" ref="file" name="filename" id="filename"
                    v-on:change="handleFileup()"
                     class="form-control m-1">

                    <div class="col-md-6 offset-md-3 text-center" v-if="preview">
                        <img v-bind:src="preview" class="img img-thumbnail">
                     </div>

                    <button v-if="!loading" class="btn btn-outline-danger m-1">글쓰기</button>
                    <button v-else class="btn btn-danger m-1">Loading...</button>
                </form>
            </div>
        </div>
    </div>
</template>
<script>
    export default {
        data(){
            return {
                board:{
```

BoardWrite.vue

```
</template>
<script>
    export default {
        data(){
            return {
                board:{idx:1, name:'', subject:'', content:'', filename:'',},
                preview:'', //업로드 이미지 미리보기시 사용
                loading:false,
            }
        },
        computed:{
            //store의 데이터들은 computed에서 가져올 수 있다.
            result(){   return this.$store.state.result;   },
        },
        methods:{
            //이미지 미리보기 처리 메소드----------------
            handleFileup(){
                this.board.filename=this.$refs.file.files[0];
                var file =this.board.filename; //e.target.files[0];
                if (file && file.type.match(/^image\/(png|jpeg)$/)) {
                    this.preview = window.URL.createObjectURL(file)
                    //업로드 이미지 미리 보여주기
                }
            },
            handleSubmit(e){//글쓰기 처리------------
                e.preventDefault();
                this.$store.dispatch('doWrite', this.board);
            }
        },
        watch:{
            result:function(val){ //result값이 변할 때 호출된다.
                //alert('val='+val);
                if(val){
                    alert('글 등록 성공');
                    this.$store.dispatch('doList');//글목록 가져오는 액션 dispatch
```

BoardWrite.vue

```
163            handleSubmit(e){//글쓰기 처리------------
164                e.preventDefault();
165                this.$store.dispatch('doWrite', this.board);
166            }
167        },
168        watch:{
169            result:function(val){ //result값이 변할 때 호출된다.
170                //alert('val='+val);
171                if(val){
172                    alert('글 등록 성공');
173                    this.$store.dispatch('doList');//글목록 가져오는 액션 dispatch
174                    /* //vuex를 사용하지 않는다면 이벤트를 발생시켜 전달해야 함
175                    this.$emit('write-ok');//부모 컴포넌트에 write-ok란 이벤트를 발생시킨다.
176                    그러면 부모 컴포넌트는 write-ok를 v-on으로 수신하여
177                    그에 따라 처리할 함수(getBoardData())를 호출한다.*/
178                }
179            }
180        }
181    }
182 </script>
183
184 <style lang="scss" scoped>
185 </style>
186
```

실습

- [1] BoardList.vue, BoardEdit.vue 등을 작성해보자.
- [2] store/index.js 파일을
- store/index.js, board_state.js, mutations.js, actions.js 등으로 분리하여 코드 관리를 해보자.

- Vuex구조 예

```
├─ index.html
├─ main.js
├─ api
│  └─ ... # API 요청을 위한 추상화를 포함합니다.
├─ components
│  ├─ App.vue
│  └─ ...
└─ store
   ├─ index.js          # 모듈을 조합하고 저장소를 내보내는 곳 입니다.
   ├─ actions.js        # 루트 액션
   ├─ mutations.js      # 루트 변이
   └─ modules
      ├─ cart.js        # cart 모듈
      └─ products.js    # products 모듈
```

참고 사이트

- vue.js공식문서: https://vuejs.org/v2/guide/
- vue.js스타일 가이드: https://vuejs.org/v2/style-guide/
- Vue.js쿡북: https://vuejs.org/v2/cookbook/

- vuex공식문서: https://vuex.vuejs.org/
- Vuex참고blog : https://blog.martinwork.co.kr/vuejs/2017/10/01/what-is-vuex.html
- vuerouter공식문서: https://router.vuejs.org/
- vue cli공식문서: https://cli.vuejs.org/

- Ajax테스트 데이터 서비스 사이트: https://jsonplaceholder.typicode.com/users