KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

pinia를 이용한 상태 관리

[KB] IT's Your Life

KB 국민은행

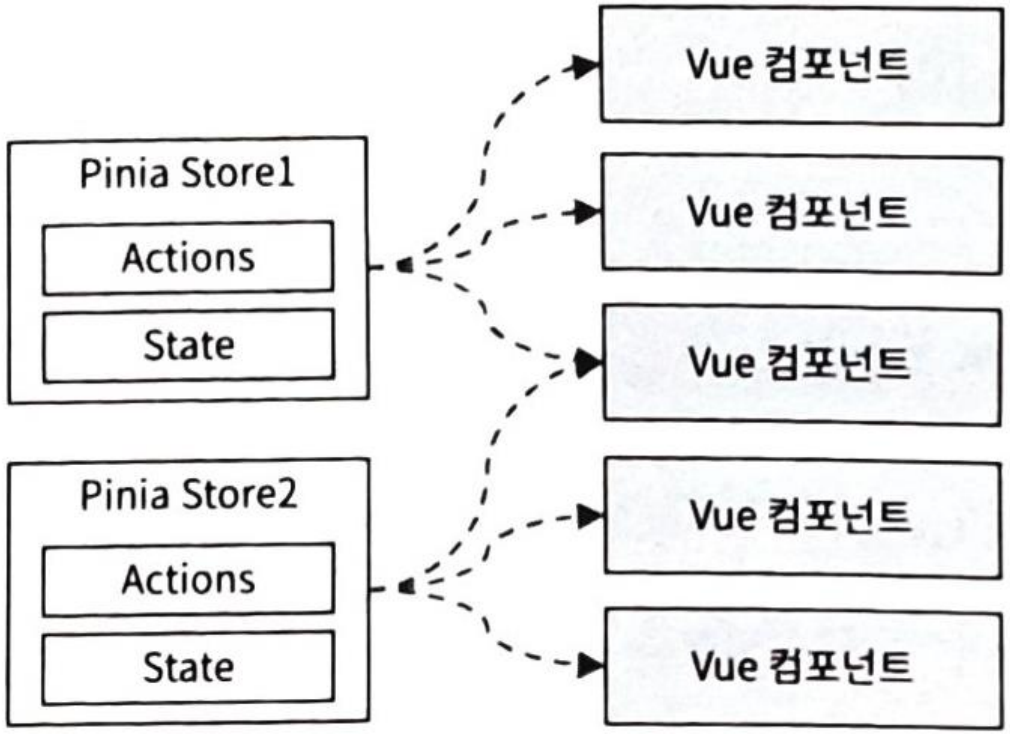
1 pinia란?

pinia

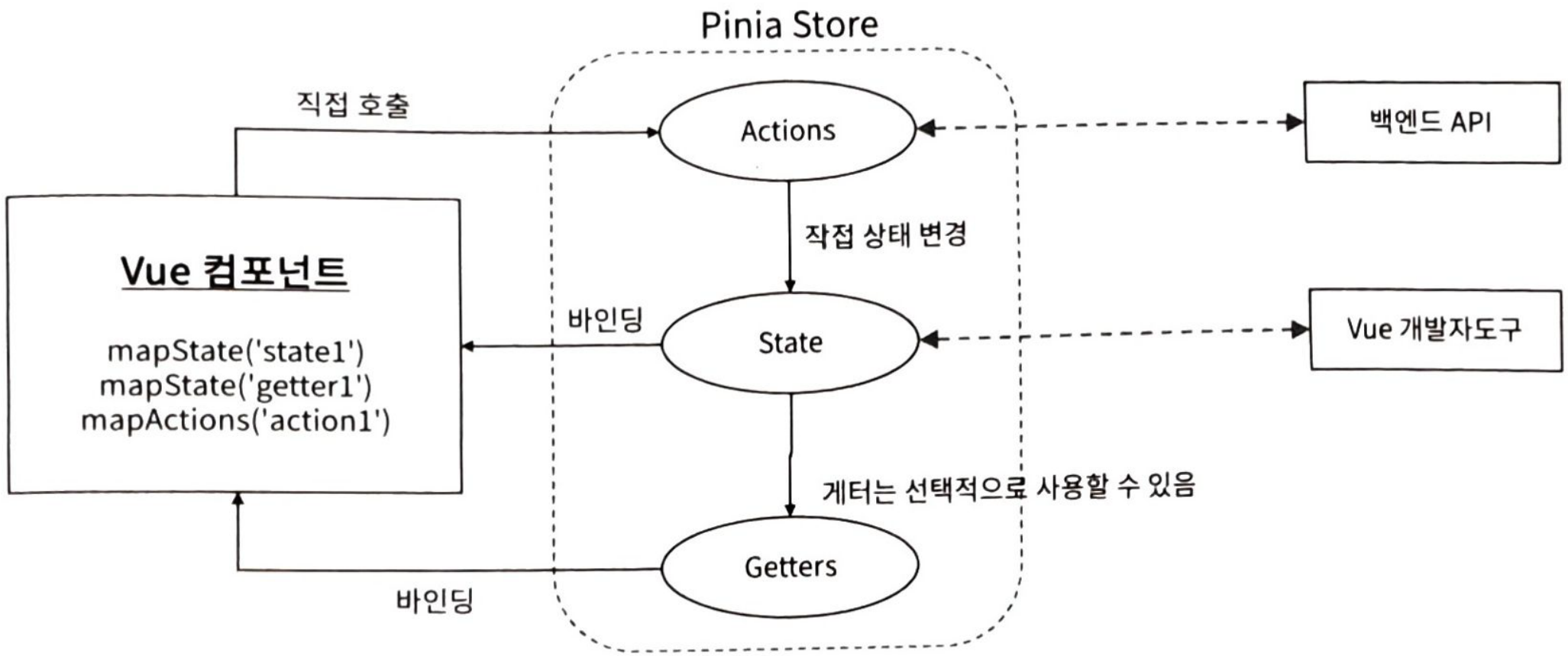
- Composition API 방식으로 Vue 애플리케이션을 위한 중앙 집중화된 상태관리 기능을 제공
- Vue3의 공식 상태 관리 라이브러리
  - 프로젝트 생성시 추가할지 질문에 yes 답변하면 자동 추가

- 참고
  - 이전에는 vuex라는 상태 관리 라이브러리 사용

Pinia Store


Pinia Store1
Actions
State
Pinia Store2
Actions
State
Vue 컴포넌트
Vue 컴포넌트
Vue 컴포넌트
Vue 컴포넌트
Vue 컴포넌트

pinia 아키텍처


Pinia Store
직접 호출
Actions
백엔드 API
작접 상태 변경
Vue 컴포넌트
mapState('state1')
mapState('getter1')
mapActions('action1')
바인딩
State
Vue 개발자도구
게터는 선택적으로 사용할 수 있음
Getters
바인딩

스토어 정의

defineStore 함수 이용

- defineStore('스토어명', 함수)

함수 인자

- 반응형 상태 정의
- 계산된 상태 정의
- 상태에 접근하는 action 함수 정의

- 외부에서 사용할 항목을 객체로 리턴

```
// [ 컴포지션 API 방법 적용 ]
export const useCount2Store = defineStore('count2', ()=>{
    const state = reactive({ count : 0 });
    const increment = ({ num }) => {
        state.count +=num;
    }
    const count = computed(()=>state.count);

    return { count, increment };
})
```

pinia를 사용하도록 Vue 애플리케이션 설정

- 프로젝트 생성시 pinia 추가하면 자동 생성됨

```
import { createApp } from 'vue'
import { createPinia } from 'pinia'
import App from './App.vue'

const pinia = createPinia()

const app = createApp(App)

app.use(pinia)
app.mount('#app')
```

컴포넌트에서 스토어 사용

- 스토어 import
- setup에서 반응성 있게 연결

[컴포지션 API를 사용한 컴포넌트에서의 스토어 사용]

```
<script>
import { useCount1Store} from '@/store/counter.js'
import { computed } from 'vue';

export default {
  setup() {
    const store = useCount1Store();
    const count =  computed(()=>store.count);
    const increment = store.increment;

    return { count, increment };
  }
}
</script>
```

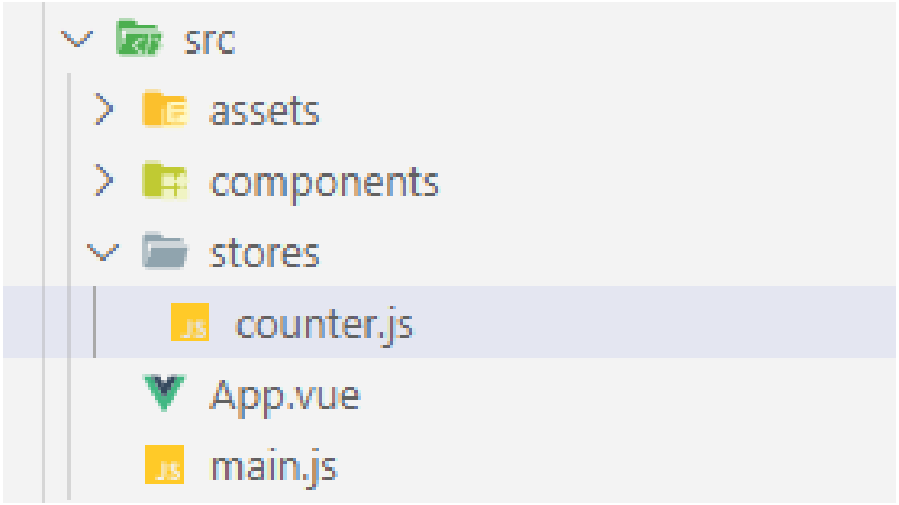
3 간단한 pinia 예제 작성

프로젝트 만들기

npm init vue pinia-test-app

Vue.js - The Progressive JavaScript Framework

✓ Add TypeScript? ... No / Yes
✓ Add JSX Support? ... No / Yes
✓ Add Vue Router for Single Page Application development? ... No / Yes
✓ Add Pinia for state management? ... No / Yes
✓ Add Vitest for Unit Testing? ... No / Yes
✓ Add an End-to-End Testing Solution? » No
✓ Add ESLint for code quality? ... No / Yes



src/main.js

```
import './assets/main.css'

import { createApp } from 'vue'
import { createPinia } from 'pinia'
import App from './App.vue'

const app = createApp(App)

app.use(createPinia())

app.mount('#app')
```

src/stores/counter.js

```
import { ref, computed } from 'vue'
import { defineStore } from 'pinia'

export const useCounterStore = defineStore('counter', () => {
  const count = ref(0)
  const doubleCount = computed(() => count.value * 2)
  function increment() {
    count.value++
  }

  return { count, doubleCount, increment }
})
```

todoList 스토어 생성

- src/stores/todoList.js
- 반응형 상태
  - state { todoList: [] }

- 액션
  - addTodo(todo)
  - deleteTodo(id)
  - toggleDone(id)

- 계산된 상태
  - doneCount → 리턴값이 기본 타입인 경우 사용자 측에서도 계산된 속성으로 연결해야 함!
  - todoList → 참조형인 경우 바로 사용 가능

src/stores/todoList.js

```
export const useTodoListStore = defineStore("todoList", ()=> {
    // 방응형 상태
    const state = reactive({
        todoList : [
            { id: 1, todo: "ES6학습",  done: false },
            { id: 2, todo: "React학습",  done: false },
            { id: 3, todo: "ContextAPI 학습",  done: true },
            { id: 4, todo: "야구경기 관람", done: false },
        ]
    })

    // action
    const addTodo = (todo) => {
        state.todoList.push({ id: new Date().getTime(), todo, done: false })
    }

    const deleteTodo = (id) => {
        let index = state.todoList.findIndex((todo) => todo.id === id);
        state.todoList.splice(index, 1);
    }

    const toggleDone =  (id) => {
        let index = state.todoList.findIndex((todo) => todo.id === id);
        state.todoList[index].done = !state.todoList[index].done;
    }
```

src/stores/todoList.js

```
    // 계산된 속성
    const doneCount = computed(()=> {
        return state.todoList.filter((todoItem)=>todoItem.done === true).length;
    })

    const todoList = computed(()=> state.todoList);

    return { todoList, doneCount, addTodo, deleteTodo, toggleDone };
})
```

src/App.vue

```
<template>
  <div>
    <h2>TodoList 테스트(Composition API)</h2>
    <hr />
    할일 추가 :
    <input type="text" v-model="todo" />
    <button @click="addTodoHandler">추가</button>
    <hr />
    <ul>
      <li v-for="todoItem in todoList">
        <span style="cursor: pointer" @click="toggleDone(todoItem.id)">
          {{ todoItem.todo }} {{ todoItem.done ? '(완료)' : '' }}
        </span>
           
        <button @click="deleteTodo(todoItem.id)">삭제</button>
      </li>
    </ul>
    <div>완료된 할일 수 : {{ doneCount }}</div>
  </div>
</template>
```

src/App.vue

```
<script setup>
import { useTodoListStore } from '@/stores/todoList.js';
import { ref, computed } from 'vue';

const todo = ref('');

const todoListStore = useTodoListStore();
const { todoList, addTodo, deleteTodo, toggleDone } = todoListStore;

const doneCount = computed(() => todoListStore.doneCount);// 기본 타입에 대해서는 계산된 속성을 다시 작성

const addTodoHandler = () => {
  addTodo(todo.value);
  todo.value = '';
};
</script>
```

3 간단한 pinia 예제 작성