KB금융그룹 | 국민의 평생 금융파트너

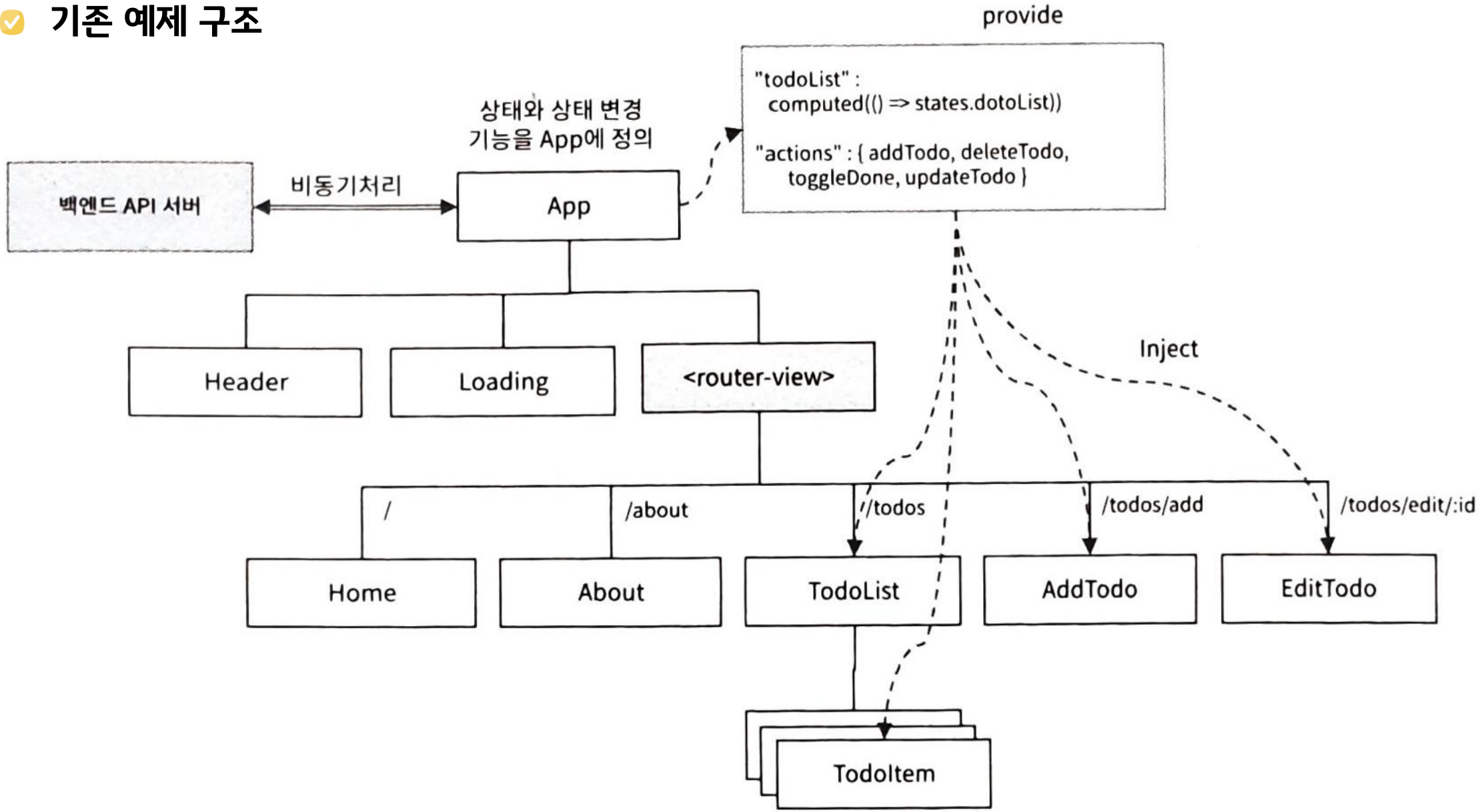
2025년 상반기 K-디지털 트레이닝

# todolist 예제에 pinia 적용하기

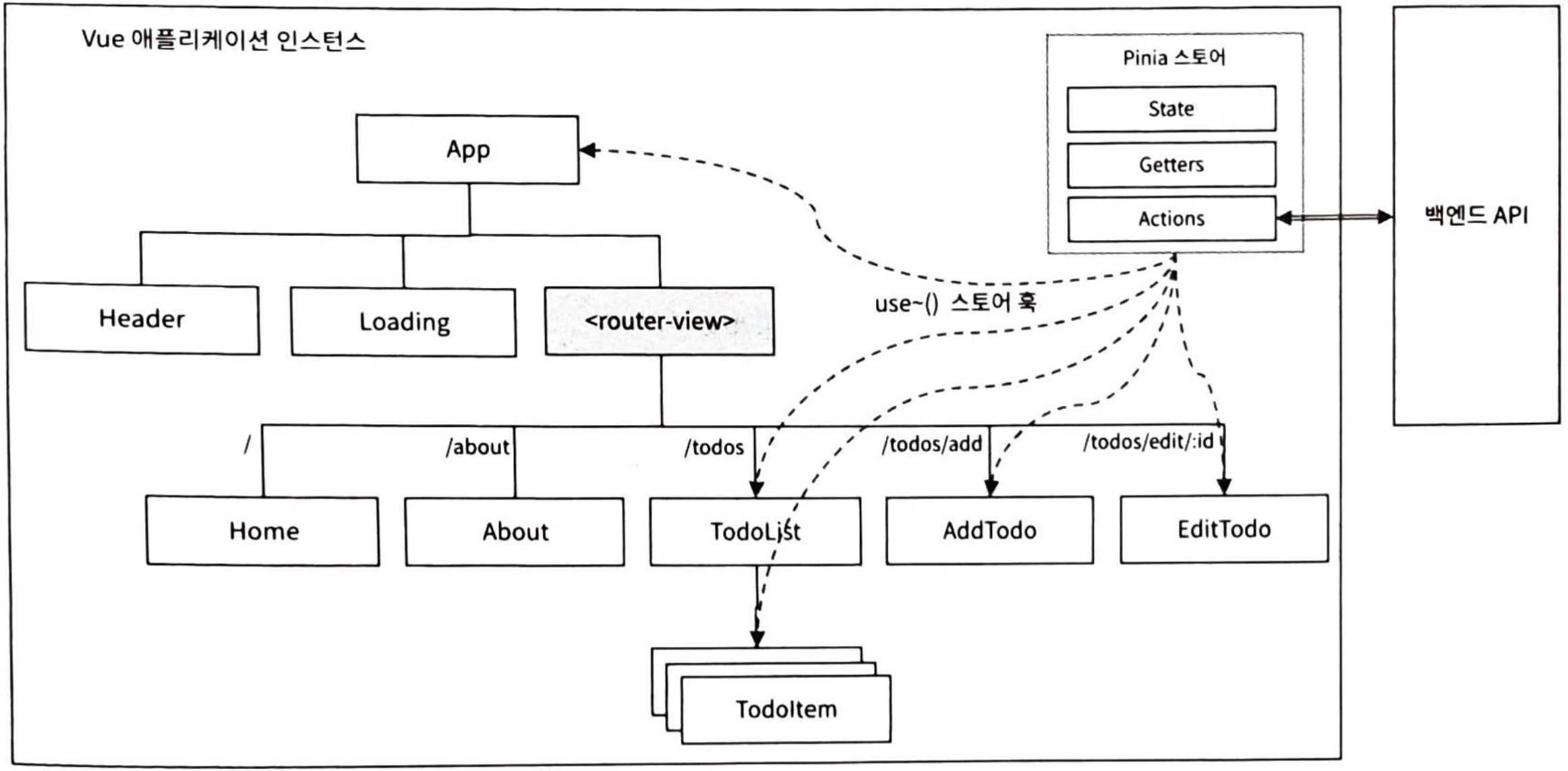
[KB] IT's Your Life

KB 국민은행

## 기존 예제 구조


provide
"todoList" :
computed(() => states.dotoList))
"actions" : { addTodo, deleteTodo,
toggleDone, updateTodo }
상태와 상태 변경
기능을 App에 정의
백엔드 API 서버
비동기처리
App
Header
Loading
<router-view>
Inject
/
/about
/todos
/todos/add
/todos/edit/:id
Home
About
TodoList
AddTodo
EditTodo
TodoItem

## 새롭게 변경할 pinia를 적용한 예제 구조



# 1 기존 예제 구조 검토

- pinia 추가
  - npm i pinia

## src/main.js 변경

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

import { createApp } from 'vue';
import App from './App.vue';
import router from './router';
import 'bootstrap/dist/css/bootstrap.css';

import { createPinia } from 'pinia';

const pinia = createPinia();
const app = createApp(App);
app.use(pinia);
app.use(router);
app.mount('#app')
```

# 2 스토어 작성

## src/stores/todoList.js 추가

```
import { defineStore } from "pinia";
import { reactive, computed } from 'vue';
import axios from 'axios';


export const useTodoListStore = defineStore('todoList', () => {
  const BASEURI = '/api/todos';
  const state = reactive({ todoList: [] });
```

## src/stores/todoList.js 추가

```
//TodoList 목록을 조회합니다.
const fetchTodoList = async () => {
  try {
    const response = await axios.get(BASEURI);
    if (response.status === 200) {
      state.todoList = response.data;
    } else {
      alert('데이터 조회 실패');
    }
  } catch (error) {
    alert('에러발생 :' + error);
  }
};
```

## src/stores/todoList.js 추가

```
// 새로운 TodoItem을 추가합니다.
const addTodo = async ({ todo, desc }, successCallback) => {
  try {
    const payload = { todo, desc };
    const response = await axios.post(BASEURI, payload);
    if (response.status === 201) {
      state.todoList.push({ ...response.data, done: false });
      successCallback();
    } else {
      alert('Todo 추가 실패');
    }
  } catch (error) {
    alert('에러발생 :' + error);
  }
};
```

## src/stores/todoList.js 추가

```
// 기존 TodoItem을 변경합니다.
const updateTodo = async ({ id, todo, desc, done }, successCallback) => {
  try {
    const payload = { id, todo, desc, done };
    const response = await axios.put(BASEURI + `/${id}`, payload);
    if (response.status === 200) {
      let index = state.todoList.findIndex((todo) => todo.id === id);
      state.todoList[index] = payload;
      successCallback();
    } else {
      alert('Todo 변경 실패');
    }
  } catch (error) {
    alert('에러발생 :' + error);
  }
};
```

## src/stores/todoList.js 추가

```
//기존 TodoItem을 삭제합니다.
const deleteTodo = async (id) => {
  try {
    const response = await axios.delete(BASEURI + `/${id}`);

    if (response.status === 200) {
      let index = state.todoList.findIndex((todo) => todo.id === id);
      state.todoList.splice(index, 1);
    } else {
      alert('Todo 삭제 실패');
    }
  } catch (error) {
    alert('에러발생 :' + error);
  }
};
```

## src/stores/todoList.js 추가

```
//기존 TodoItem의 완료여부(done) 값을 토글합니다.
const toggleDone = async (id) => {
  try {
    let todo = state.todoList.find((todo) => todo.id === id);
    let payload = { ...todo, done: !todo.done };
    const response = await axios.put(BASEURI + `/${id}`, payload);
    if (response.status === 200) {
      todo.done = payload.done;
    } else {
      alert('Todo 완료 변경 실패');
    }
  } catch (error) {
    alert('에러발생 :' + error);
  }
};
```

## src/stores/todoList.js 추가

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

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

## src/App.vue 변경

```
<template>
  <div class="container">
      <Header />
      <router-view />
      <Loading v-if="isLoading" />
  </div>
</template>

<script setup>
import { computed } from 'vue';
import Header from '@/components/Header.vue'
import { useTodoListStore } from '@/stores/todoList.js'
import Loading from '@/components/Loading.vue'

const todoListStore = useTodoListStore();
const isLoading = computed(()=>todoListStore.isLoading);
const fetchTodoList = todoListStore.fetchTodoList;
fetchTodoList();
</script>
```

# 3 TodoList, TodoItem 컴포넌트 변경

## src/pages/TodoList.vue 변경

```
<template>
    …
    <div class="row">
   …
        <span>완료된 할일 : {{doneCount}}</span>
    </div>
</template>

<script setup>
import { computed } from 'vue';
import { useTodoListStore } from '@/stores/todoList.js'
import TodoItem from '@/components/TodoItem.vue'

const todoListStore = useTodoListStore();
const { fetchTodoList } = todoListStore;
const doneCount = computed(()=>todoListStore.doneCount);
const todoList = computed(()=>todoListStore.todoList);
</script>
```

## src/components/TodoItem.vue 변경

```
<template>
  …
</template>

<script setup>
import { useRouter } from 'vue-router';
import { useTodoListStore } from '@/stores/todoList.js'

defineProps({
  todoItem: { Type: Object, required:true }
})

const router = useRouter();
const todoListStore = useTodoListStore();
const { deleteTodo, toggleDone } = todoListStore;
</script>
```

# 4 AddTodo, EditTodo 컴포넌트 변경

## src/pages/AddTodo.vue 변경

```
…
<script setup>
import { reactive } from 'vue';
import { useRouter } from 'vue-router';
import { useTodoListStore } from '@/stores/todoList.js'

const router = useRouter();
const { addTodo } = useTodoListStore();
const todoItem =  reactive({ todo:"", desc:"" })

const addTodoHandler = () => {
    let { todo } = todoItem;
    if (!todo || todo.trim()==="") {
        alert('할일은 반드시 입력해야 합니다');
        return;
    }
    addTodo({ ...todoItem }, ()=>{
        router.push('/todos')
    });
}
</script>
```

## src/pages/EditTodo.vue 변경

```
…
<script setup>
import { reactive } from 'vue';
import { useRouter, useRoute } from 'vue-router';
import { useTodoListStore } from '@/stores/todoList.js'

const router = useRouter();
const currentRoute = useRoute();

const { todoList, updateTodo, } = useTodoListStore();

const matchedTodoItem = todoList.find((item)=> item.id === currentRoute.params.id);
if (!matchedTodoItem)  {
    router.push('/todos');
}

const todoItem =  reactive({ ...matchedTodoItem });
```

## src/pages/EditTodo.vue 변경

```
const updateTodoHandler = () => {
    let { todo } = todoItem;
    if (!todo || todo.trim()==="") {
        alert('할일은 반드시 입력해야 합니다');
        return;
    }
    updateTodo({ ...todoItem }, ()=>{
        router.push('/todos');
    });
}
</script>
```