

2024년 상반기 K-디지털 트레이닝

로그인

[KB] IT's Your Life



- ☑ API 기반 Spring Security에서 처리 함
 - 추가 작업 없음

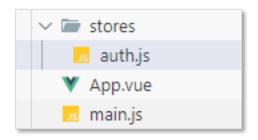


2024년 상반기 K-디지털 트레이닝

인증 스토어

[KB] IT's Your Life





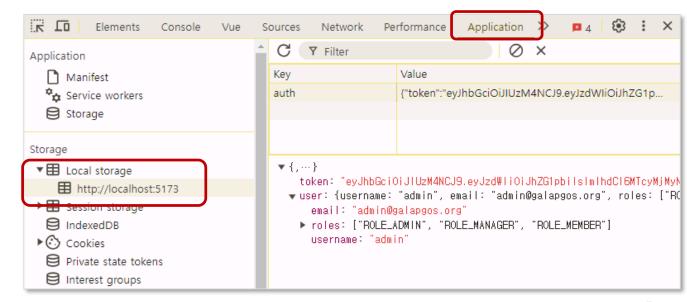
- 서버 인증은 추후에 작업
- 임시 작업 로그인하면 무조건 로그인 되는 것으로
- 새로 고침을 하면 스토어가 리셋 됨
 - 상태가 변경될 때(로그인, 로그아웃)마다 localStorage에 저장
 - 첫 기동할 때 localStorage에서 상태 복원

localStorage

- 웹 어플리케이션 운영가능하는 저장소
- <키, 값>의 쌍으로 정보를 저장
 - 키, 값 모두 문자열로 처리
 - 객체 정보를 저장할 때 JSON 문자열로 변환해서 저장

ㅇ 주요 메서드

- setItem(키, 값)
 - 값을 JSON 직렬화하여 문자열로 저장
- getItem(키)
 - 리턴된 문자열을 JSON 역직렬화로 객체 복원
- removeltem(₹|)
- clear()

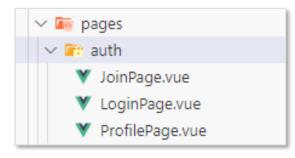


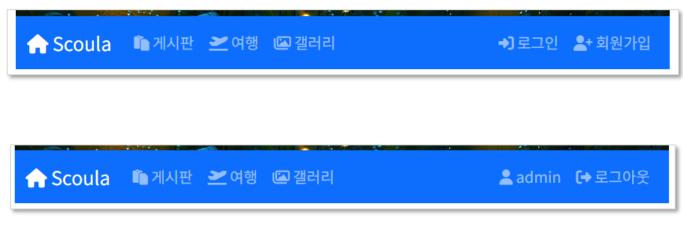
```
import { ref, computed, reactive } from 'vue';
import { defineStore } from 'pinia';
const initState = {
 token: '', // 접근 토큰(JWT)
 user: {
   username: '', // 사용자 ID
   email: '', // Email
   roles: [], // 권한 목록
 },
};
export const useAuthStore = defineStore('auth', () => {
 const state = ref({ ...initState });
 const isLogin = computed(() => !!state.value.user.username); // 로그인 여부
 const username = computed(() => state.value.user.username); // 로그인 사용자 ID
 const email = computed(() => state.value.user.email); // 로그인 사용자 email
```

```
const login = async (member) => {
  state.value.token = 'test token';
  state.value.user = { username: member.username, email: member.username + '@test.com' } ;
  localStorage.setItem('auth', JSON.stringify(state.value));
};
const logout = () => {
  localStorage.clear();
  state.value = { ...initState };
};
const getToken = () => state.value.token;
```

```
const load = () => {
    const auth = localStorage.getItem('auth');
    if (auth != null) {
        state.value = JSON.parse(auth);
        console.log(state.value);
    }
};
load();
return { state, username, email, isLogin, logout, getToken };
});
```

🥑 로그인 페이지







```
<script setup>
import { computed, reactive, ref } from 'vue';
import { useAuthStore } from '@/stores/auth';
import { useRouter } from 'vue-router';
const router = useRouter();
const auth = useAuthStore();
const member = reactive({
 username: '',
 password: '',
});
const error = ref('');
const disableSubmit = computed(() => !(member.username && member.password));
```

```
const login = async () => {
 console.log(member);
 try {
   await auth.login(member);
   router.push('/');
 } catch (e) {
   // 로그인 에러
   console.log('에러=====', e);
   error.value = e.response.data;
</script>
```

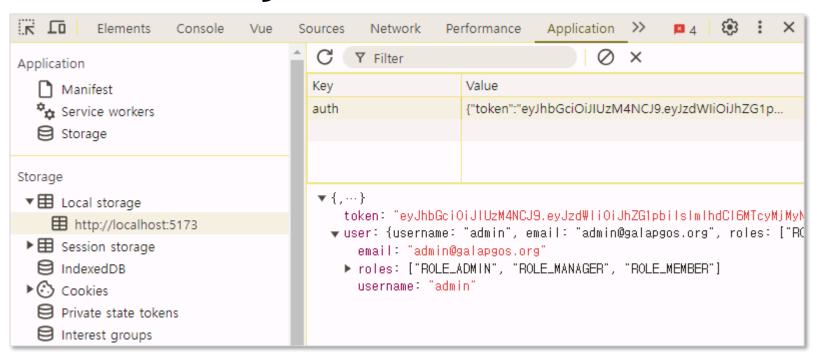
```
<template>
 <div class="mt-5 mx-auto" style="width: 500px">
    <h1 class="my-5">
     <i class="fa-solid fa-right-to-bracket"></i>
     로그인
    </h1>
    <form @submit.prevent="login">
     <div class="mb-3 mt-3">
        <label for="username" class="form-label">
          <i class="fa-solid fa-user"></i></i></or>
         사용자 ID:
        </label>
        <input type="text" class="form-control" placeholder="사용자 ID" v-model="member.username" />
     </div>
```

```
<div class="mb-3">
       <label for="password" class="form-label">
         <i class="fa-solid fa-lock"></i>
         비밀번호:
       </label>
       <input type="password" class="form-control" placeholder="비밀번호" v-model="member.password"/>
     </div>
     <div v-if="error" class="text-danger">{{ error }}</div>
     <button type="submit" class="btn btn-primary mt-4" :disabled="disableSubmit">
       <i class="fa-solid fa-right-to-bracket"></i></i>
       로그인
     </button>
   </form>
 </div>
</template>
```

AccountMenuGroup.vue

```
<script setup>
const { login, join } = config.accoutMenus;
import { useAuthStore } from '@/stores/auth.js';
const auth = useAuthStore();
const islogin = computed(() => auth.isLogin);
const username = computed(() => auth.username);
</script>
<template>
</template>
```

🦁 로그인 후 localStorage 확인



☑ 로그아웃

- o auth 스토어의 logout 호출
 - localStorage 클리어
 - 상태를 초기 상태로 설정

✓ LogoutMenultem.vue

```
<script setup>
import { useRouter } from 'vue-router';
import { useAuthStore } from '@/stores/auth';
const store = useAuthStore();
const router = useRouter();
const logout = (e) => {
 // 로그아웃
 store.logout();
  router.push('/');
};
</script>
<template>
 <a href="#" class="nav-link" @click.prevent="logout">
   <i class="fa-solid fa-right-from-bracket"></i></i>
    로그아웃
 </a>
</template>
```

Storage	Key	Value
▼ ■ Local storage		
http://localhost:5173		
▶ ■ Session storage		
☐ IndexedDB		
► Cookies		
Private state tokens		
☐ Interest groups		
▶ Shared storage		
Cache storage		
Storage buckets		



2024년 상반기 K-디지털 트레이닝

로그인 - axios 인터셉터

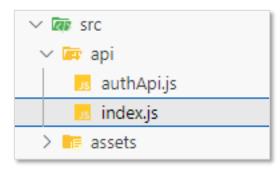
[KB] IT's Your Life



axios 인터셉터(interceptor)

- o axios의 요청과 응답에 대한 필터 역할
 - 모든 요청을 가로채서 요청을 수정할 수 있음.
 - 모든 응답을 가로채서 응답을 수정할 수 있음
- backend와의 모든 통신은 인증 토큰(jwt)를 포함해야 함
- o Axios Interceptor 운영해야 함
 - 자동으로 모든 요청의 request 헤더에 인증 헤더 추가
 - Authentication: Bearer <토큰 문자열>
- 인터셉터가 설정된 axios 인스턴스를 작성하고, 백엔드 통신시 이 인스턴스를 사용

axios 인터셉터(interceptor)



☑ api/index.js (기본 골격)

```
import axios from 'axios';
const instance = axios.create({
 timeout: 1000,
});
// 요청 인터셉터
instance.interceptors.request.use(
  (config) => {
   // config.headers : 요청 헤더
   return config;
 },
  (error) => {
   // 요청 중 에러가 난 경우
   return Promise.reject(error);
);
```

☑ api/index.js (기본 골격)

```
// 응답 인터셉터
instance.interceptors.response.use(
 (response) => {
   // 정상 응답인 경우 (200, 404)
   return response;
 },
 async (error) => {
   // 에러 응답인 경우(401, 403, 305, 500 등)
   return Promise.reject(error);
export default instance; // 인터셉터가 적용된 axios 인스턴스
```

💟 요청 인터셉터에서 할일

- o auth store에서 토큰을 추출
- 토큰이 있다면 요청 헤더에 추가
 - Authentication: Bearer <토큰 문자열>

○ 응답 인터셉터에서 할일

- 401 에러인 경우
 - 로그인 페이지로 이동

☑ api/index.js

```
import axios from 'axios';
import { useAuthStore } from '@/stores/auth';
import router from '@/router';

const instance = axios.create({
   timeout: 1000,
});
```

api/index.js

```
// 요청 인터셉터
instance.interceptors.request.use(
  (config) => {
   // JWT 추출
   const { getToken } = useAuthStore();
   const token = getToken();
   if (token) {
     // 토큰이 있는 경우
     config.headers['Authorization'] = `Bearer ${token}`;
     console.log(config.headers.Authorization);
   return config;
 },
  (error) => {
   return Promise.reject(error);
);
```



```
// 응답 인터셉터
instance.interceptors.response.use(
  (response) => {
   if (response.status === 200) {
     return response;
    if (response.status === 404) {
     return Promise.reject('404: 페이지 없음 ' + response.request);
    return response;
 },
 async (error) => {
   if (error.response?.status === 401) {
     const { logout } = useAuthStore();
     logout();
     router.push('/auth/login?error=login_required');
     return Promise.reject({ error: '로그인이 필요한 서비스입니다.' });
   return Promise.reject(error);
export default instance;
```

- ☑ 실제 api 서버로 로그인하기
 - o axios를 이용해서 '/api/auth/login' 요청

```
import { ref, computed, reactive } from 'vue';
import { defineStore } from 'pinia';
import axios from 'axios';
export const useAuthStore = defineStore('auth', () => {
  const login = async (member) => {
   // state.value.token = 'test token';
    // state.value.user = { username: member.username, email: member.username + '@test.com' } ;
   // api 호출
    const { data } = await axios.post('/api/auth/login', member);
    state.value = { ...data };
    localStorage.setItem('auth', JSON.stringify(state.value));
 };
});
```

localStorage 확인





2024년 상반기 K-디지털 트레이닝

로그인 - 아바타 처리

[KB] IT's Your Life



아바타 처리(백엔드)

UploadFiles.java

```
package org.scoula.common.util;
public class UploadFiles {
    public static void downloadImage(HttpServletResponse response, File file) {
        try {
            Path path = Path.of(file.getPath());
            String mimeType = Files.probeContentType(path);
            response.setContentType(mimeType);
            response.setContentLength((int) file.length());
            try (OutputStream os = response.getOutputStream();
                BufferedOutputStream bos = new BufferedOutputStream(os)) {
                Files.copy(path, bos);
        catch (Exception e) {
            throw new RuntimeException(e);
```

아바타 처리 (백엔드)

MemberController.java

```
public class MemberController {
   @GetMapping("/{username}/avatar")
   public void getAvatar(@PathVariable String username, HttpServletResponse response) {
       String avatarPath = "c:/upload/avatar/" + username + ".png";
       File file = new File(avatarPath);
       if(!file.exists()) { // 아바타 등록이 없는 경우, 디폴트 아바타 이미지 사용
           file = new File("C:/upload/avatar/unknown.png");
       UploadFiles.downloadImage(response, file);
```

아바타 처리 (클라이언트)

assets/main.css

```
.avatar {
 border-radius: 50%;
 border-color: gray;
 background-color: #d8d8d8;
.avatar-sm {
 width: 20px;
 height: 20px;
.avatar-md {
 width: 40px;
 height: 40px;
.avatar-lg {
 width: 80px;
 height: 80px;
```

아바타 처리 (클라이언트)

AccountMenuItem.vue

```
<script setup>
const props = defineProps({ username: String });
                                                                                         ② user0 → 로그아웃
const avatar = `/api/member/${props.username}/avatar`;
</script>
<template>
 <router-link class="nav-link" to="/auth/profile">
     <img :src="avatar" class="avatar avatar-sm" />
     {{ username }}
   </router-link>
 </template>
<style></style>
```