

**실습 - 로그인**

**my memo**

# Backend

- <https://github.com/kgbsibbi/miniNodeServer>
- 로그인, 메모 작성, 메모 리스트 조회 등의 기능을 제공하는 Node.js Server
- sqlite를 사용해 데이터를 저장하기 때문에 별도의 데이터베이스 설치 필요 없다.

# Server API

Url	Method	Request	Response	
/	GET			
/admin/users	GET		[[ userid:string, name:string, password:string]]	
/admin/users/{userid}	GET		{ userid:string, name:string, password:string}	특정 사용자 정보 조회
/admin/users/{userid}	DELETE			사용자 삭제
/admin/memos	GET		[[ memoid:int, userid:string title:string, content:string, fileUrl:string, originalFileName:string, savedTime:int ]]	모든 사용자가 작성한 메모 리스트 조회
/users/	POST	Body {userid:string, name:string, password:string}		회원 가입-아이디/이름/비번
/users/{userid}	GET	Header {authorization: token}	{userid:string, name:string, password:string}	로그인한 사용자가 자신의 정보 조회
/users/{userid}	DELETE	Header {authorization: token}		로그인한 사용자가 자신의 정보 삭제
/auth/login	POST	Body {usreid:string, password:string}	{token:string, name:string}	아이디와 비밀번호를 이용한 로그인
/auth/autologin	POST	Header {authorization: token}	{userid:string, name:string}	자동 로그인. 토큰 체크

# Server API

Url	Method	Request	Response	
/memos	GET	Header {authorization: token}	[[ memoid:int, userid:string title:string, content:string, fileUrl:string, originalFileName:string, savedTime:int ]]	자신의 메모 목록 조회
/memos	POST	Header {authorization: token} Body: <b>Multi-part request</b> {title:string, content:string, file:file, originalFileName:string}		메모 추가 -파일 첨부 포함
/memos/{memoid}	GET	Header {authorization: token}	[[ memoid:int, userid:string title:string, content:string, fileUrl:string, originalFileName:string, savedTime:int ]]	자신의 메모 1개 조회
/memos/{memoid}	PUT	Header {authorization: token} Body: <b>Multi-part request</b> {title:string, content:string, file:file, originalFileName:string }		자신의 메모 업데이트
/memos/{memoid}	DELETE			자신의 메모 삭제

프로젝트 생성

# 프로젝트 생성

- vue create my-memo
- cd my-memo
- vue add router
- vue add vuetify
- npm install --save axios
- npm install --save vuex

# 프로젝트 생성

- vue.config.js file에 proxy 내용을 추가

```
1  module.exports = {  
2    transpileDependencies: [  
3      'vuetify'  
4    ],  
5    devServer: {  
6      proxy: {  
7        '/api': {  
8          target: 'http://localhost:3000',  
9          changeOrigin: true,  
10         pathRewrite: {  
11           '^/api': ''  
12         }  
13       }  
14     }  
15   }  
16 }
```



# views

- views 폴더에 다음 항목들 추가
  - Home.vue (기존 내용 삭제), AddMemo.vue, ReadMemo.vue
  - Signin.vue, Signup.vue

```
1  <template>
2  |  <div></div>
3  </template>
4
5  <script>
6  |  export default {
7  |    name: 'Home'
8  |  }
9  </script>
```

# router

- router/index.js
  - about은 삭제

```
1  import Vue from 'vue'
2  import VueRouter from 'vue-router'
3  import Home from '../views/Home.vue'
4  import Add from '../views/AddMemo.vue'
5  import Read from '../views/ReadMemo.vue'
6  import Signin from '../views/Signin'
7  import Signup from '../views/Signup'
8
9  Vue.use(VueRouter)
10
11  const routes = [
12    { path: '/', name: 'Home', component: Home},
13    { path: '/add', name: 'Add', component: Add},
14    { path: '/memos/:memoId', name: 'Read', component: Read},
15    { path: '/signin', name: 'Signin', component: Signin},
16    { path: '/signup', name: 'Signup', component: Signup},
17  ]
18
19  const router = new VueRouter({
20    mode: 'history',
21    base: process.env.BASE_URL,
22    routes
23  })
24
25  export default router
```

# vue router: navigation guard

- 특정 주소를 요청할 때 미리 확인하여 다음 중 하나로 처리
  - 허용: 원하는 주소로 이동
  - 리다이렉션: 다른 주소로 이동
  - 취소: 이동 하지 않음

# vue router: navigation guard

## 전역 가드

- 모든 라우팅 동작마다 체크를 하는 방법
- router/index.js의 new VueRouter 함수 호출 다음에 설정 가능

```
21  const router = new VueRouter({
22    mode: 'history',
23    base: process.env.BASE_URL,
24    routes
25  })
26
27  router.beforeEach((to, from, next)=>{
28    next();
29  });
```

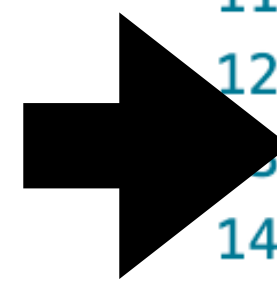
예시

# vue router: navigation guard

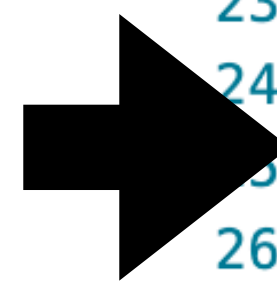
## 전역 가드

- /memos는 로그인이 필요할 때 다음과 같이 구성 가능

- router/index.js



```
11 const routes = [  
12   { path: '/', name: 'Home', component: Home},  
13   { path: '/add', name: 'Add', component: Add, meta:{requiresAuth:true}}},  
14   { path: '/memos/:memoId', name: 'Read', component: Read},  
15   { path: '/signin', name: 'Signin', component: Signin},  
16   { path: '/signup', name: 'Signup', component: Signup},  
17 ]  
  
18  
19 const router = new VueRouter({  
20   mode: 'history',  
21   base: process.env.BASE_URL,  
22   routes  
23 })  
  
24  
25 router.beforeEach((to, from, next)=>{  
26   if(to.matched.some((record)=>record.meta.requiresAuth)){  
27     alert('Signin please')  
28     next('/signin');  
29   } else {  
30     next();  
31   }  
32 });
```



# vue router: navigation guard

## 라우트 별 가드

- 특정 주소에만 가드를 설정하는 방법

```
11  const routes = [  
12    { path: '/', name: 'Home', component: Home},  
13    { path: '/add', name: 'Add', component: Add, meta:{requiresAuth:true}},  
14    { path: '/memos/:memoId',  
15      name: 'Read', component: Read,  
16      beforeEnter:(to, from, next)=>{  
17        alert('Signin please')  
18        next('/signin');  
19      }  
20  },  
21  { path: '/signin', name: 'Signin', component: Signin},  
22  { path: '/signup', name: 'Signup', component: Signup},  
23  ]
```



# vue router: navi

## 컴포넌트 내부 가드

- 컴포넌트에서 체크
- Home.vue

```
1 <template>
2   <div>
3     <p v-if="requiresAuth">
4       <router-link :to="{ path: '/signin' }">Signin</router-link>
5     </p>
6   </div>
7 </template>
8
9 <script>
10   export default {
11     name: 'Home',
12     data(){
13       return {
14         requiresAuth:false
15       }
16     },
17     beforeRouteEnter(to, from, next){
18       next(vm=>{
19         vm.requiresAuth=true
20       })
21     },
22     beforeRouteLeave(to, from, next){
23       this.requiresAuth=false
24       next()
25     }
26   }
27 </script>
```

# vue router: navigation guard

## 코드 정리

- router/index.js
- 전역 가드로 통일. 다음과 같이 파일 수정하고 Home.vue에 있는 코드도 삭제



```
11 const routes = [  
12   { path: '/', name: 'Home', component: Home, meta:{requiresAuth:true}},  
13   { path: '/add', name: 'Add', component:Add, meta:{requiresAuth:true}},  
14   { path: '/memos/:memoId', name:'Read', component:Read, meta:{requiresAuth:true}},  
15   { path: '/signin', name:'Signin', component:Signin},  
16   { path: '/signup', name:'Signup', component:Signup},  
17 ]  
18  
19 const router = new VueRouter({  
20   mode: 'history',  
21   base: process.env.BASE_URL,  
22   routes  
23 })  
24  
25 router.beforeEach((to, from, next)=>{  
26   if(to.matched.some((record)=>record.meta.requiresAuth)){  
27     alert('Signin please')  
28     next('/signin');  
29   } else {  
30     next();  
31   }  
32 });
```



# vuex

- 로그인 할 때 서버가 발행한 토큰을 저장해야 한다.
- 로그아웃 하거나 서버가 토큰 만료를 알리면 토큰을 삭제해야 한다.
- src 폴더에 store 폴더를 만들고 그 안에 index.js 파일 생성
- 새로 고침을 해도 데이터가 날아가지 않도록 localStorage에 저장한다.

# store/index.js

```
1  import Vue from 'vue'
2  import Vuex from 'vuex'
3  import axios from 'axios'
4
5  Vue.use(Vuex)
6
7  export default new Vuex.Store({
8    state:{
9      accessToken:null
10   },
11   getters:{
12     isAuth(state){
13       if(state.accessToken==null) return false;
14       return true;
15     }
16   },
17   mutations:{
18     signin(state, payload){
19       state.accessToken = payload.accessToken
20       localStorage.setItem('accessToken',state.accessToken)
21     },
22     signout(state){
23       state.accessToken=null
24       localStorage.removeItem('accessToken')
25       location.reload();
26     },
27     getAccessToken(state){
28       state.accessToken = localStorage.getItem('accessToken')
29     }
30   },
31   actions:{
32     signin({commit}, payload){
33       const data={ userid:payload.userid, password:payload.password}
34       return axios.post('/api/auth/login', data)
35         .then(response=>{
36           if(response.status==200){//로그인 성공
37             commit('signin', {accessToken:response.data.token})
38           }
39         })
40         .catch(()=>{ // 에러 발생하면 로그아웃 처리
41           commit('signout')
42         })
43     }
44   }
45 });
```

# main.js

```
1  import Vue from 'vue'
2  import App from './App.vue'
3  import router from './router'
4  import vuetify from './plugins/vuetify'
5  import store from './store'
6
7  Vue.config.productionTip = false
8
9  new Vue({
10   router,
11   vuetify,
12   store,
13   beforeCreate() {
14     this.$store.commit('getAccessToken')
15   },
16   render: h => h(App),
17 }).$mount('#app')
```



# router/index.js

```
1  import Vue from 'vue'
2  import VueRouter from 'vue-router'
3  import Home from '../views/Home.vue'
4  import Add from '../views/AddMemo.vue'
5  import Read from '../views/ReadMemo.vue'
6  import Signin from '../views/Signin'
7  import Signup from '../views/Signup'
8
9  Vue.use(VueRouter)
10
11  const auth=(to, from, next)=>{
12    if(to.matched.some((record)=>record.meta.requiresAuth)){
13      if(localStorage.getItem('accessToken')==null){
14        alert('Signin please')
15        next('/signin');
16      }
17    }
18    next();
19  }
```



# Signin.vue

```
1  <template>
2    <div>
3      <h1>Signin</h1>
4      <form @submit.prevent="onSubmit(userid, password)">
5        <input type="text" v-model="userid" placeholder="User ID" />
6        <input type="password" v-model="password" placeholder="Password" />
7        <input type="submit" value="Signin" />
8      </form>
9      <p><i>{{message}}</i></p>
10    </div>
11  </template>
```

```
13  <script>
14    export default {
15      name: 'Signin',
16      data(){
17        return {
18          userid: '',
19          password: '',
20          message: ''
21        }
22      },
23      methods:{
24        onSubmit(userid, password){
25          this.$store.dispatch('signin', {userid, password})
26            .then(()=>{
27              this.$router.push('/')
28            })
29            .catch(()=>{
30              this.message='Signin Failed.'
31            })
32        }
33      }
34    }
35  </script>
```

# axios - auth

- header 항목 중 'Authorization' 항목에 토큰을 보내야 함
- 1) request 를 보낼 때 추가해서 보내는 법
  - 우측 코드와 같은 방법

```
mounted(){  
  const config={  
    headers:{"Authorization":this.$store.state.accessToken}  
  }  
  axios.get('/api/memos', config)  
    .then(res=>{  
      this.memos = res.data  
      console.log(this.memos)  
    })  
    .catch(()=>{  
      this.$store.commit('signout')  
      this.$router.push('/signin')  
    })  
}
```

# axios - auth

- header 항목 중 'Authorization' 항목에 토큰을 보내야 함
- 2) Interceptor
  - 각 request를 보내기 전에 처리 사항을 추가할 수 있음.
  - 각 response를 받았을 때 미리 처리할 수 있음



# axios - interceptor

- src 에 apis 폴더 추가
- apis 폴더 안에 sendRequest.js 파일 생성

```
1  import axios from 'axios'
2
3  const instance=axios.create({
4  });
5
6  instance.interceptors.request.use(
7    // Request 실행 직전
8    (config)=>{
9      const token = localStorage.getItem('accessToken')
10     if(token)
11       config.headers.Authorization=token
12     else
13       console.log('No token')
14     return config;
15   },
16   // 요청 에러 처리.
17   (error)=>{
18     return Promise.reject(error);
19   }
20 );
21
22 instance.interceptors.response.use(
23   // http status 가 200인 경우.
24   (response)=>{
25     return response;
26   },
27   // http status가 200이 아닌 경우.
28   (error)=>{
29     return Promise.reject(error)
30   }
31 );
32
33 export default instance;
```



# axios - interceptor

- apis 폴더에 memos.js 파일 추가
  - header가 필요한 요청에 대해 함수 추가

```
1  import sendRequest from './sendRequest'
2
3  export default{
4    getMemos(){
5      return sendRequest({
6        url: '/api/memos',
7        method: 'get'
8      })
9    },
10   getMemo(memoId){
11     return sendRequest({
12       url: '/api/memos/'+memoId,
13       method: 'get'
14     })
15   }
16 }
```

# axios - interceptor

- Home.vue

```
9  <script>
10  import memoApi from '../apis/memos'
11
12  export default {
13    name: 'Home',
14    data(){
15      return {
16        memos: [],
17      }
18    },
19    mounted(){
20      memoApi.getMemos()
21        .then(res=>{
22          this.memos = res.data
23        })
24        .catch(()=>{
25        })
26      }
27  }
28  </script>
```

# axios - interceptor

- ReadMemo.vue

```
7 <script>
8 import memoApi from '../apis/memos'
9 export default {
10   name: 'ReadMemo',
11   data(){
12     return {
13       memo: {}
14     }
15   },
16   mounted(){
17     memoApi.getMemo(this.$route.params.memoId)
18       .then(res=>{
19         this.memo = res.data
20         console.log(this.memo)
21       })
22     .catch(()=>{
23     })
24   }
25 }
26 </script>
```

# axios - multipart/form-data

- memo 내용과 함께 파일을 업로드하려면 multipart/form-data 를 전송해야 한다.
- apois 폴더에 sendMultipartRequest.js 를 추가

# axios - multipart/form-data

- apis/sendMultipartRequest.js
  - 'Content-Type' 에 해당하는 헤더 추가

```
1  import axios from 'axios'
2
3  const instance=axios.create({
4  });
5
6  instance.interceptors.request.use(
7    // Request 실행 직전
8    (config)=>{
9      const token = localStorage.getItem('accessToken')
10     if(token)
11       config.headers.Authorization=token
12     else
13       console.log('No token')
14     config.headers['Content-Type']='multipart/form-data';
15     return config;
16   },
17   // 요청 에러 처리.
18   (error)=>{
19     return Promise.reject(error);
20   }
21 );
22
23 instance.interceptors.response.use(
24   // http status 가 200인 경우.
25   (response)=>{
26     return response;
27   },
28   // http status가 200이 아닌 경우.
29   (error)=>{
30     return Promise.reject(error)
31   }
32 );
33
34 export default instance;
```

# axios - multipart/form-data

- AddMemo.vue

```
1 <template>
2   <div>
3     <input type="text" v-model="title"><br>
4     <input type="text" v-model="content"><br>
5     <input type="file" @change="selectFile" ref="file">
6     <button @click="addMemo">Save</button>
7   </div>
8 </template>
```

```
10 <script>
11 import memoApi from '../apis/memos'
12 export default {
13   name: 'AddMemo',
14   data(){
15     return {
16       title:'',
17       content:'',
18       file:''
19     }
20   },
21   methods:{
22     selectFile(){
23       this.file = this.$refs.file.files[0]
24     },
25     addMemo(){
26       const data = new FormData();
27       data.append('title', this.title);
28       data.append('content', this.content)
29       data.append('file', this.file)
30
31       memoApi.addMemo(data)
32       .then(response=>{
33         console.log(response.status)
34         this.$router.push('/')
35       })
36       .catch(error=>{
37         console.log(error)
38         this.$router.push('/')
39       })
40     }
41   }
42 }
43 </script>
```