

2025년 상반기 K-디지털 트레이닝

회원관리 -회원가입 (백엔드)

[KB] IT's Your Life



회원 가입

errorCheck

사용자 ID : ID 중복 확인 ID 중복 체크를 하셔야 합니다.

hong3

아바타 이미지:

파일 선택

선택된 파일 없음

email

hong@gmail.com

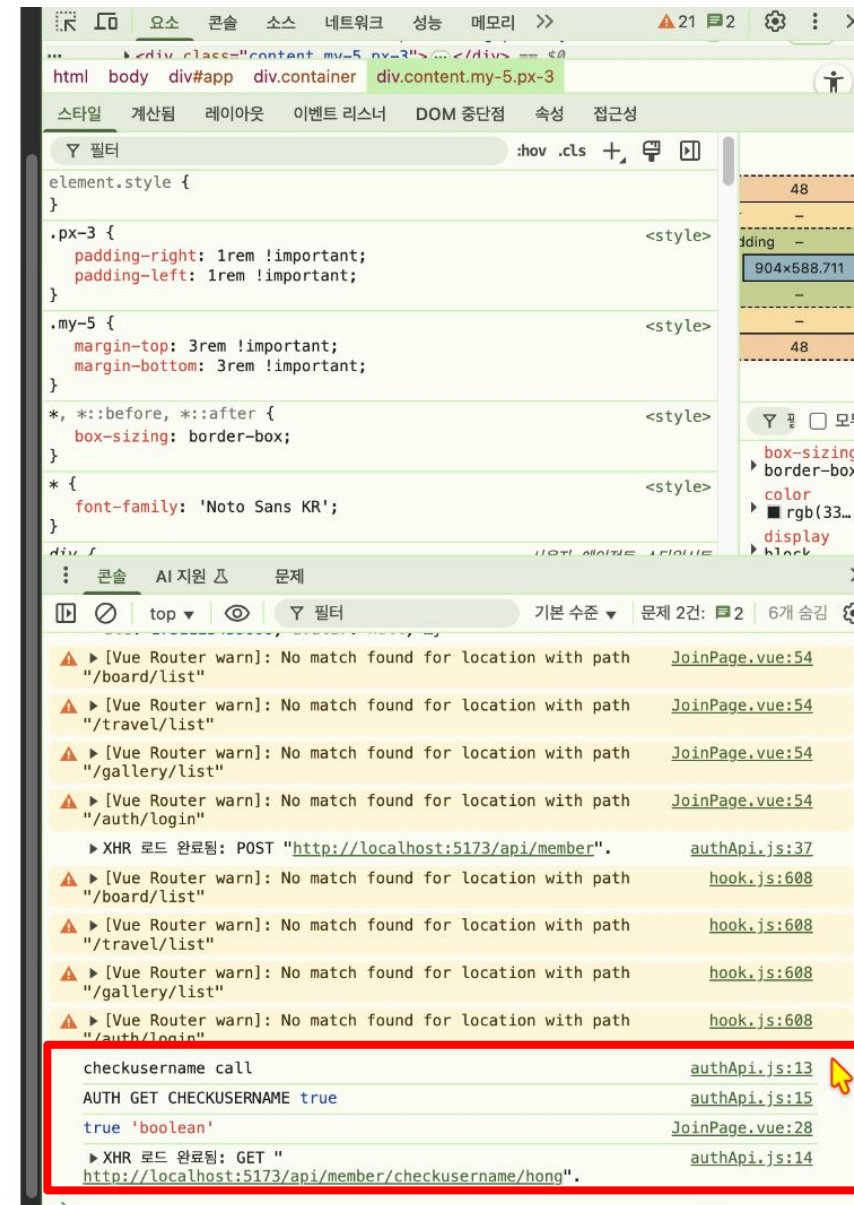
비밀번호:

....

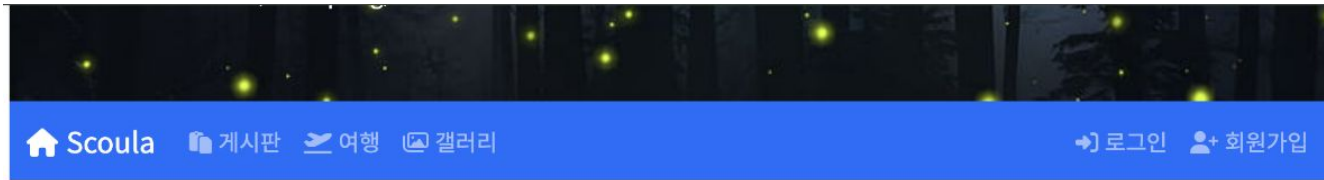
비밀번호 확인:

....

확인



ID 중복 확인 버튼을 누르면 아이디중복확인



회원 가입

사용자 ID : ID 중복 확인 이미 사용중인 ID입니다.

아바타 이미지:

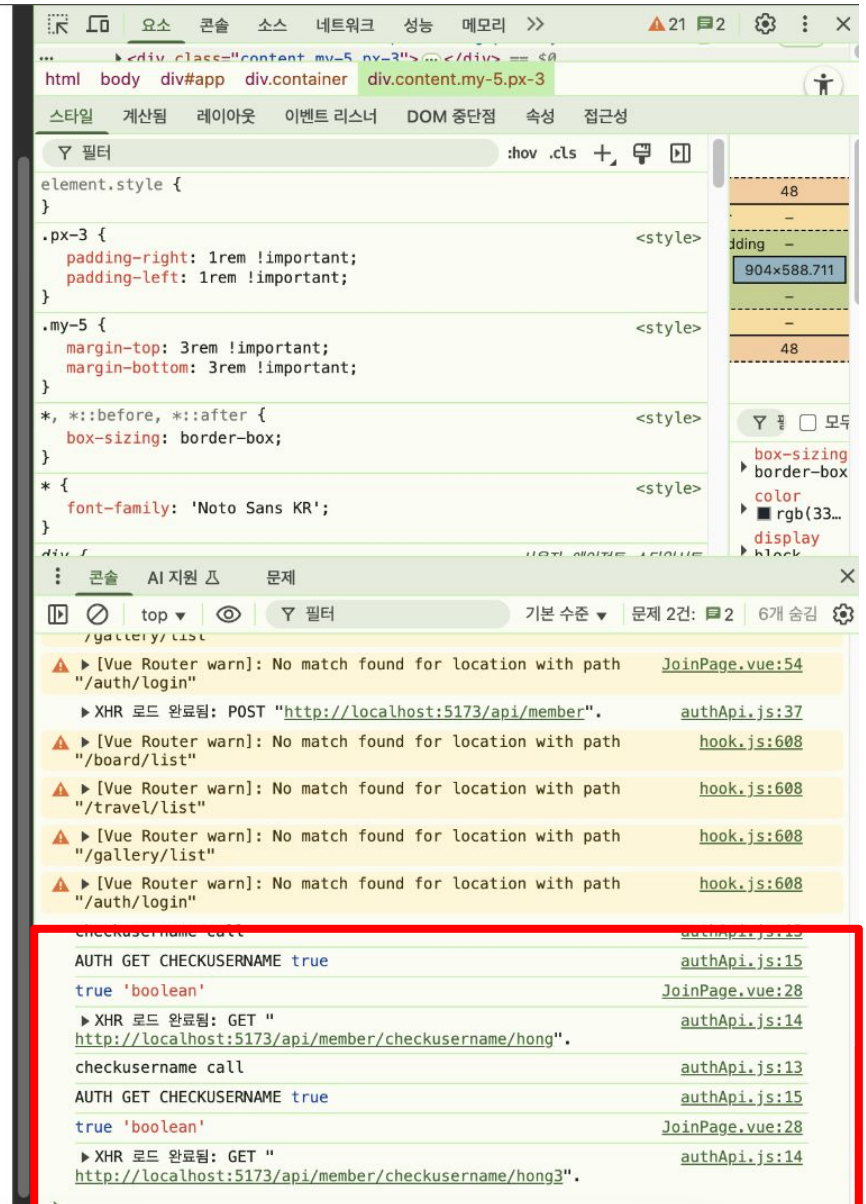
파일 선택 선택된 파일 없음

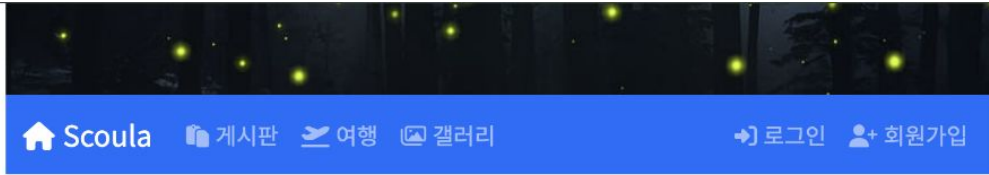
email

비밀번호:

비밀번호 확인:

확인





+ 회원 가입

사용자 ID : ID 중복 확인 사용가능한 ID입니다.

아바타 이미지:

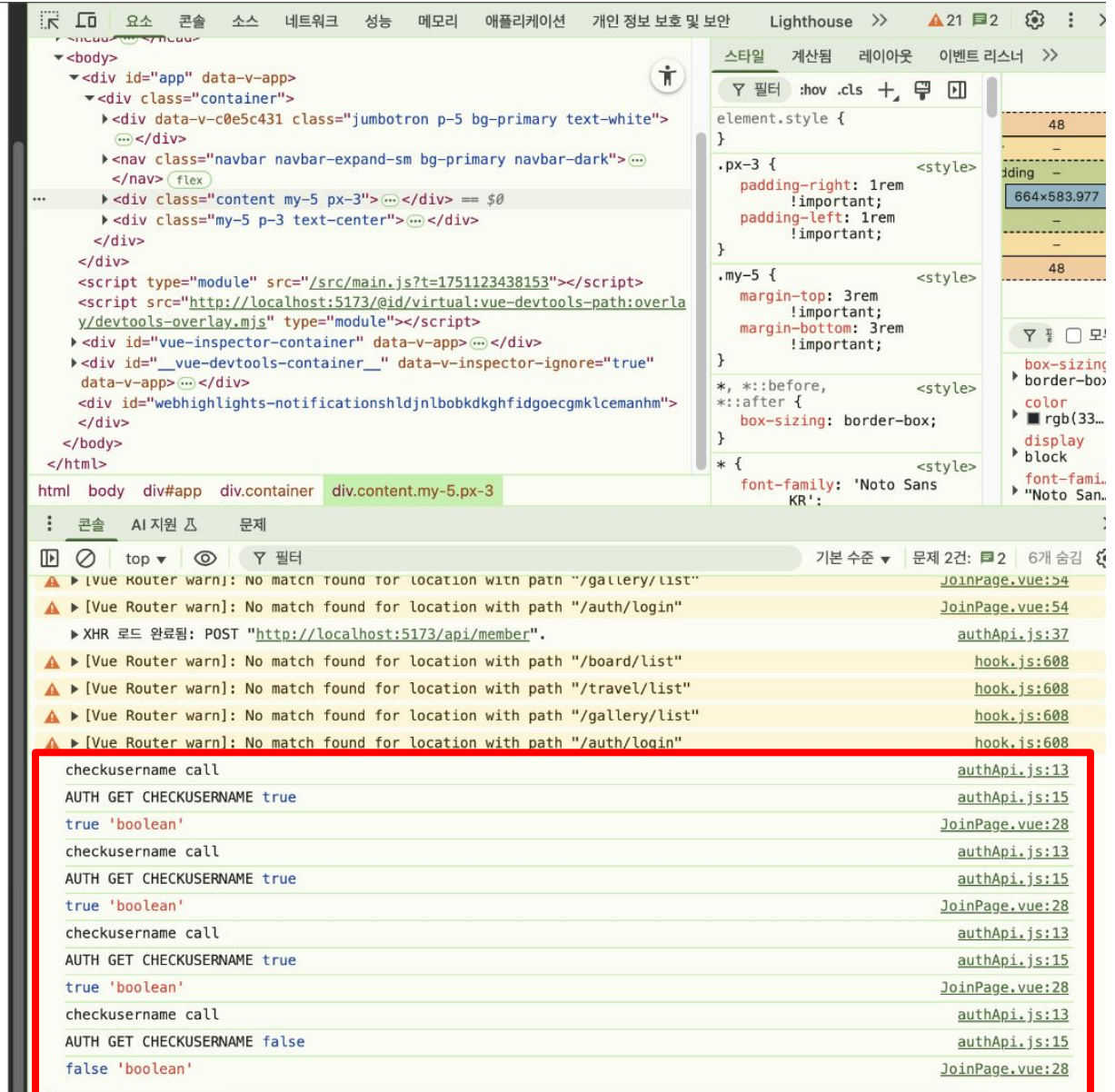
파일 선택 선택된 파일 없음

email

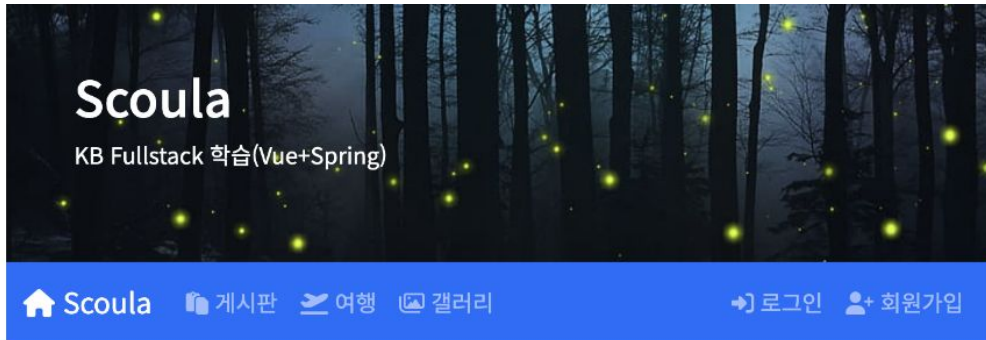
비밀번호:

비밀번호 확인:

+ 확인

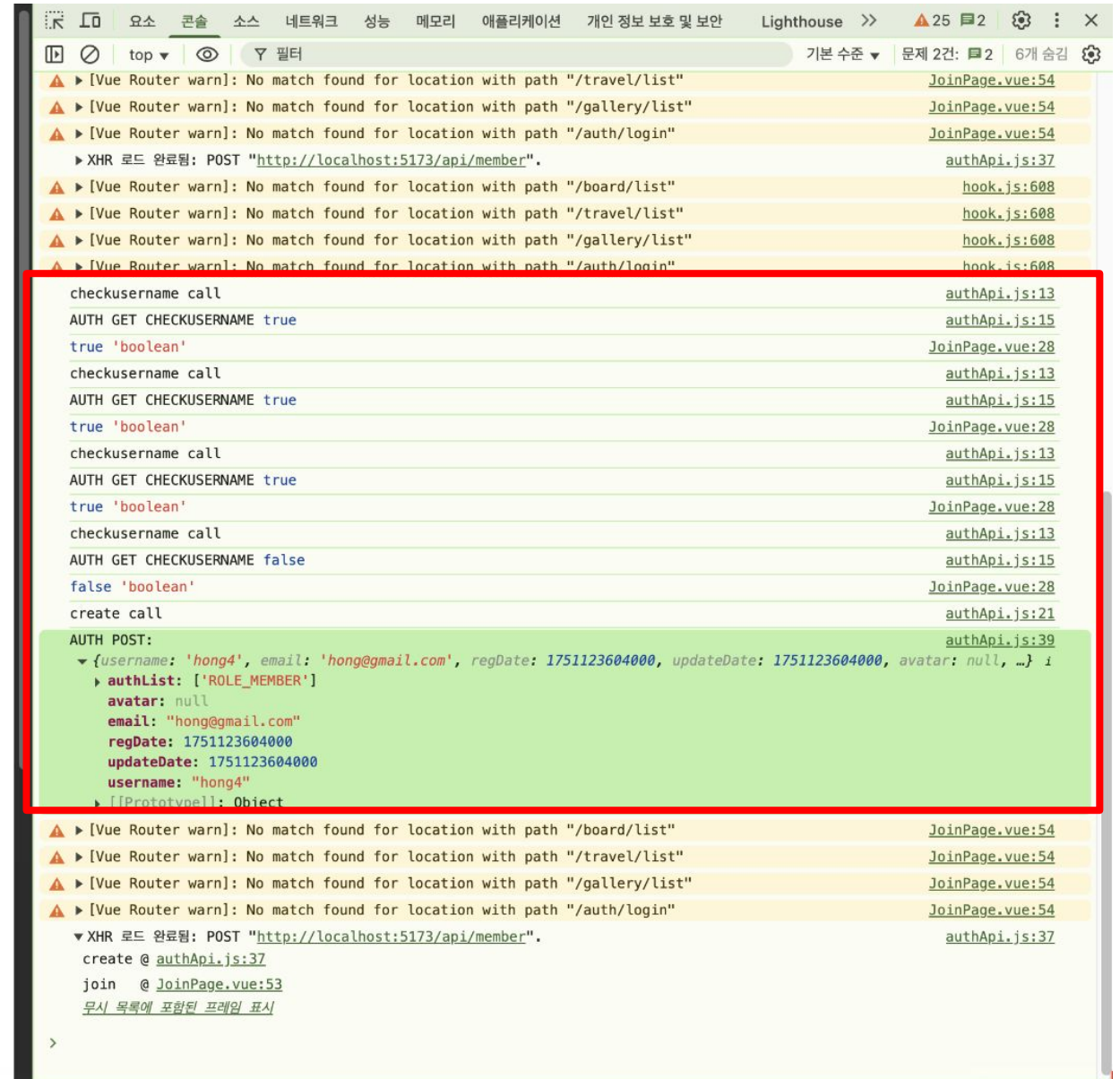
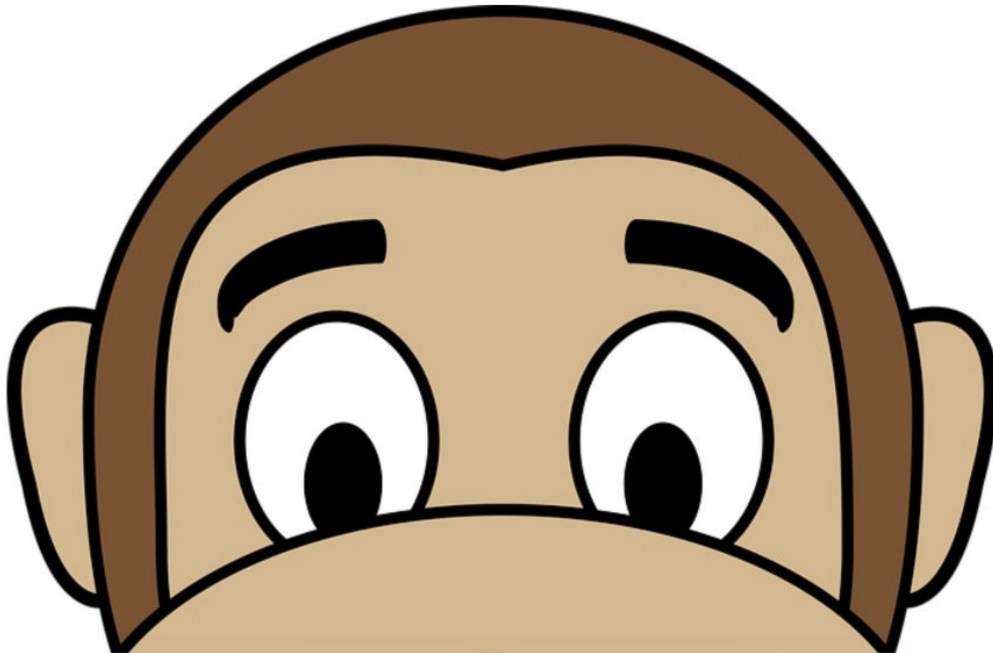


확인버튼 누르면 json으로 응답받고, /로 redirect



첫 페이지입니다.

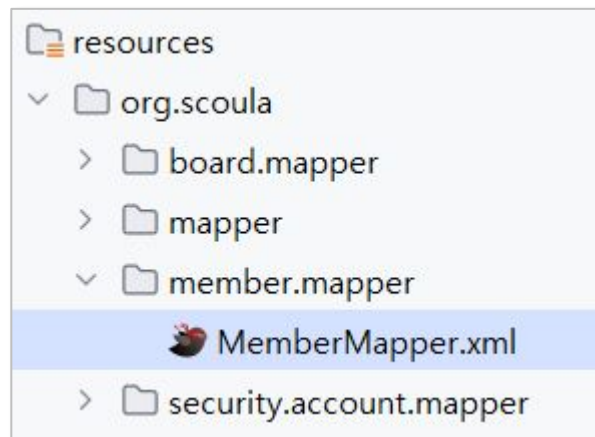
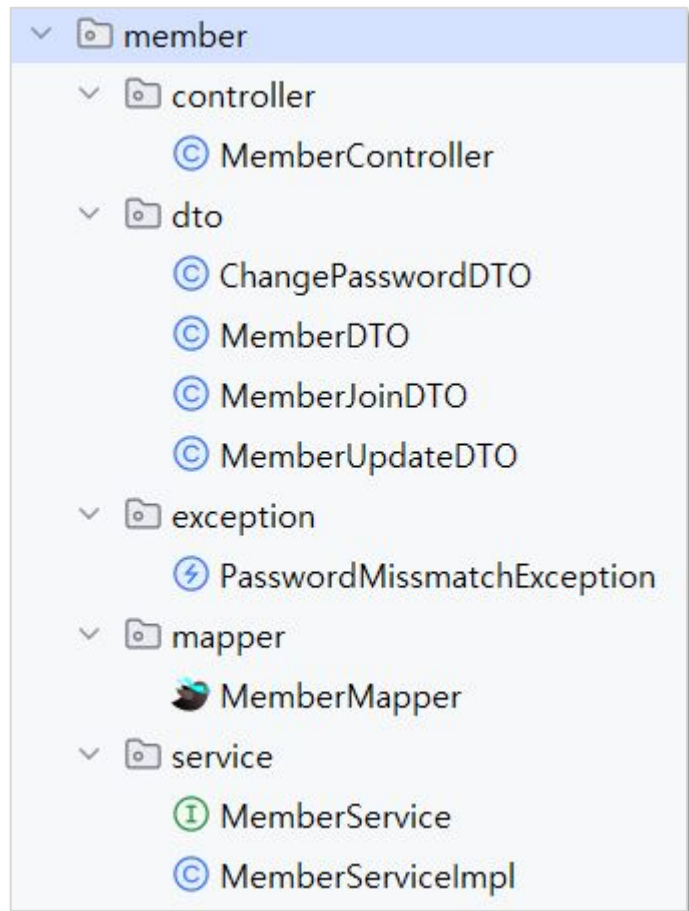
여기가 렌더링 들어가



1 회원관리 - 회원 가입

• 백엔드 준비하기

- 회원관리 기본 패키지
 - org.scoula.member



- 회원 가입

- 회원 가입시 고려사항

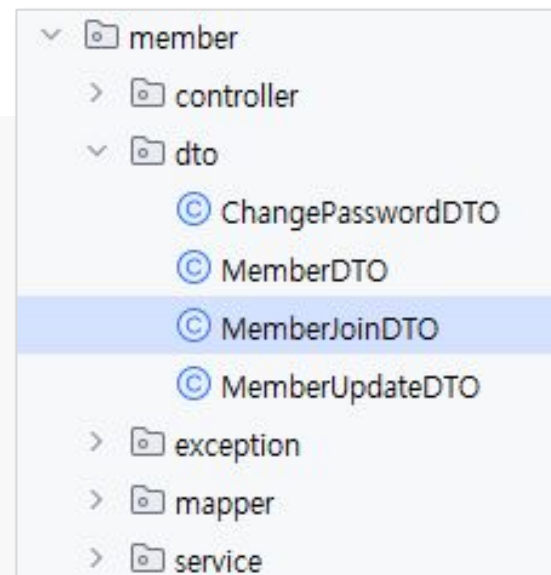
- username 중복 여부 점검
 - 비밀번호 암호화
 - 회원 정보 저장시 회원 권한도 같이 저장 □ 트랜잭션 처리
 - 회원 아바타 이미지 파일 업로드 □ c:/upload/avatar/<username>.jpg 형태로 저장

- DTO

- MemberJoinDTO → 회원가입시 사용
 - MemberDTO → 회원상세정보와 auth(role)정보 조인

• MemberJoinDTO.java

```
package org.scoula.member.dto;  
  
...  
import org.scoula.security.account.domain.MemberVO;  
import org.springframework.web.multipart.MultipartFile;  
  
@Data  
@NoArgsConstructor  
@AllArgsConstructor  
@Builder  
public class MemberJoinDTO {  
    private String username;  
    private String password;  
    private String email;  
  
    private MultipartFile avatar;
```



- **MemberJoinDTO.java**

```
public MemberVO toVO() {  
    return MemberVO.builder()  
        .username(username)  
        .password(password)  
        .email(email)  
        .build();  
}  
  
}
```

○ MemberVO에 @Builder 추가

• MemberDTO.java

```
package org.scoula.member.dto;

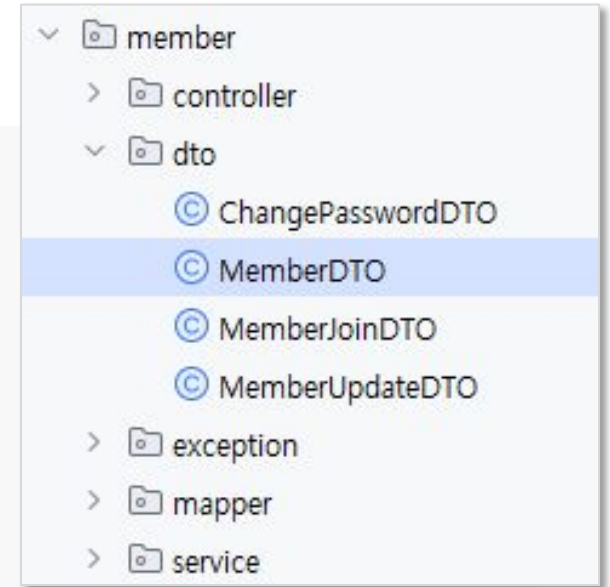
...

import org.scoula.security.account.domain.MemberVO;
import org.springframework.web.multipart.MultipartFile;
...

@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class MemberDTO {
    private String username;
    private String email;
    private Date regDate;
    private Date updateDate;

    private MultipartFile avatar;

    private List<String> authList;           // 권한 목록, join 처리 필요
```



• MemberDTO.java

```
public static MemberDTO of(MemberVO m) {  
    return MemberDTO.builder()  
        .username(m.getUsername())  
        .email(m.getEmail())  
        .regDate(m.getRegDate())  
        .updateDate(m.getUpdateDate())  
        .authList(m.getAuthList().stream().map(a->a.getAuth()).toList())  
        .build();  
}  
  
public MemberVO toVO() {  
    return MemberVO.builder()  
        .username(username)  
        .email(email)  
        .regDate(regDate)  
        .updateDate(updateDate)  
        .build();  
}  
}
```

• MemberMapper.java

```
package org.scoula.member.mapper;

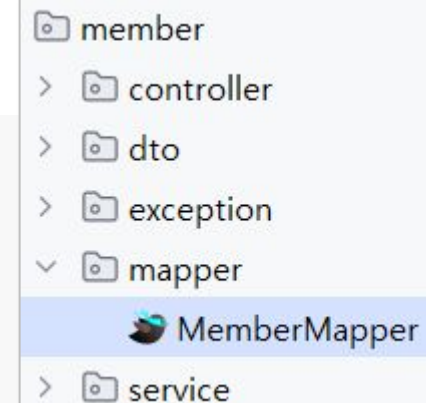
...
import org.scoula.security.account.domain.AuthVO;
import org.scoula.security.account.domain.MemberVO;

public interface MemberMapper {
    MemberVO get(String username);

    MemberVO findByUsername(String username); // id 중복 체크시 사용

    int insert(MemberVO member); // 회원 정보 추가

    int insertAuth(AuthVO auth);           // 회원 권한 정보 추가
}
```

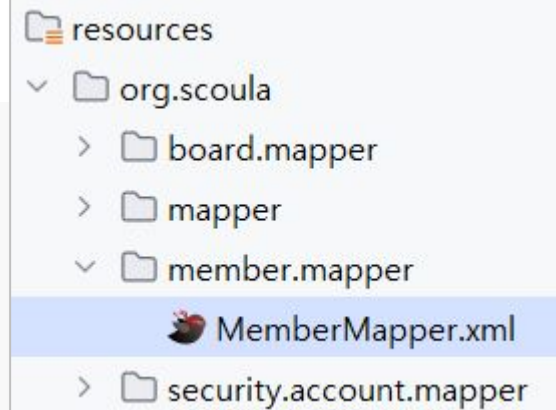


• MemberMapper.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="org.scoula.member.mapper.MemberMapper">
  <resultMap id="authMap" type="AuthVO">
    <result property="username" column="username" />
    <result property="auth" column="auth" />
  </resultMap>

  <resultMap id="memberMap" type="MemberVO">
    <id property="username" column="username" />
    <result property="username" column="username" />
    <result property="password" column="password" />
    <result property="email" column="email" />
    <result property="regDate" column="reg_date" />
    <result property="updateDate" column="update_date" />
    <collection property="authList" resultMap="authMap" />
  </resultMap>
```



- **MemberMapper.xml**

```
<select id="get" resultMap="memberMap">
    SELECT m.username, password, email, reg_date, update_date, auth
    FROM
        tbl_member m
    LEFT OUTER JOIN tbl_member_auth a
        ON m.username = a.username
    where m.username = #{username}
</select>
```

```
<select id="findByUsername" resultType="org.scoula.security.account.domain.MemberVO">
    SELECT * FROM tbl_member WHERE username = #{username}
</select>
```

- **MemberMapper.xml**

```
<insert id="insert">
  INSERT INTO tbl_member
  VALUES("#{username}", #{password}, #{email}, now(), now() )
</insert>

<insert id="insertAuth">
  INSERT INTO tbl_member_auth(username, auth)
  VALUES("#{username}", #{auth})
</insert>

</mapper>
```

• MemberService.java

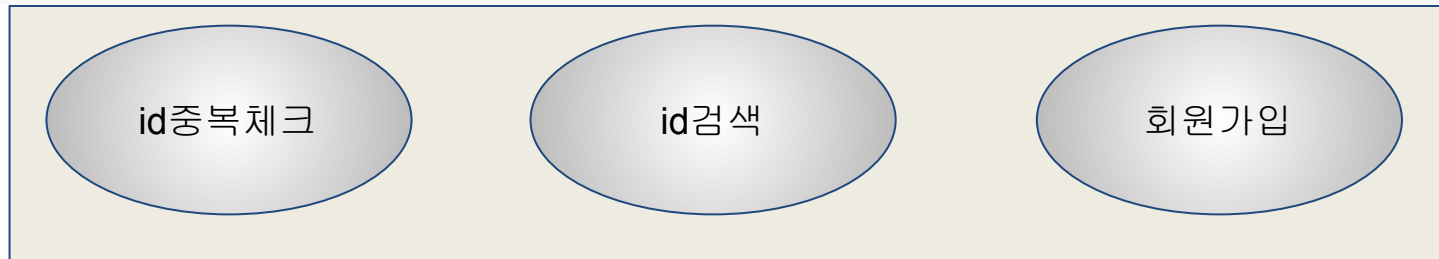
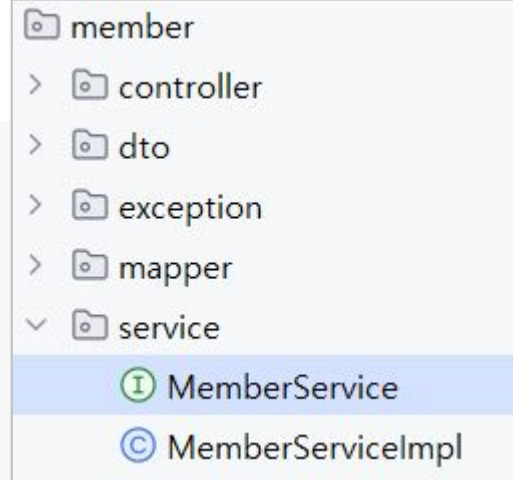
```
package org.scoula.member.service;

import org.scoula.member.dto.MemberJoinDTO;

public interface MemberService {
    boolean checkDuplicate(String username);

    MemberDTO get(String username);

    MemberDTO join(MemberJoinDTO member);
}
```



인터페이스



클래스

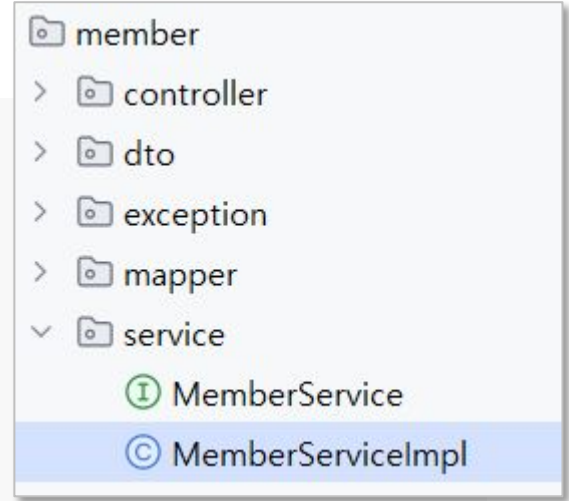
• MemberServiceImpl.java

```
package org.scoula.member.service;

...
@Log4j2
@Service
@RequiredArgsConstructor
public class MemberServiceImpl implements MemberService {
    final PasswordEncoder passwordEncoder;
    final MemberMapper mapper;

    @Override
    public boolean checkDuplicate(String username) {
        MemberVO member = mapper.findByUsername(username);
        return member != null ? true : false;
    }
}
```

id중복체크



• MemberServiceImpl.java

@Override

```
public MemberDTO get(String username) {  
    MemberVO member = Optional.ofNullable(mapper.get(username))  
        .orElseThrow(NoSuchElementException::new);  
    return MemberDTO.of(member);  
}
```

id검색

```
private void saveAvatar(MultipartFile avatar, String username) {  
    //아바타 업로드  
    if(avatar != null && !avatar.isEmpty()) {  
        File dest = new File("c:/upload/avatar", username + ".png");  
        try {  
            avatar.transferTo(dest);  
        } catch (IOException e) {  
            throw new RuntimeException(e);  
        }  
    }  
}
```

파일첨부
저장

• MemberServiceImpl.java

@Transactional

@Override

```
public MemberDTO join(MemberJoinDTO dto) {
    MemberVO member = dto.toVO();
```



```
    member.setPassword(passwordEncoder.encode(member.getPassword())); // 비밀번호 암호화
    mapper.insert(member);
```

```
    AuthVO auth = new AuthVO();
    auth.setUsername(member.getUsername());
    auth.setAuth("ROLE_MEMBER");
    mapper.insertAuth(auth);
```

```
    saveAvatar(dto.getAvatar(), member.getUsername());
```

```
    return get(member.getUsername());
```

```
}
```

```
}
```



@Transactional

→ 두 개의 작업이 모두 성공하면 db에 반영 O
→ 두 개의 작업 중 하나라도 실패하면 둘 다 db에 반영 X

- 컨트롤러
 - URL
 - GET /api/member/checkusername/{username}
 - POST /api/member

• MemberController.java

```
package org.scoula.member.controller;
```

```
...
```

```
@Log4j2
```

```
@RestController
```

```
@RequiredArgsConstructor
```

```
@RequestMapping("/api/member")
```

```
public class MemberController {
```

```
    final MemberService service;
```

```
@GetMapping("/checkusername/{username}")
```

```
public ResponseEntity<Boolean> checkUsername(@PathVariable String username) {
```

```
    return ResponseEntity.ok().body(service.checkDuplicate(username));
```

```
}
```

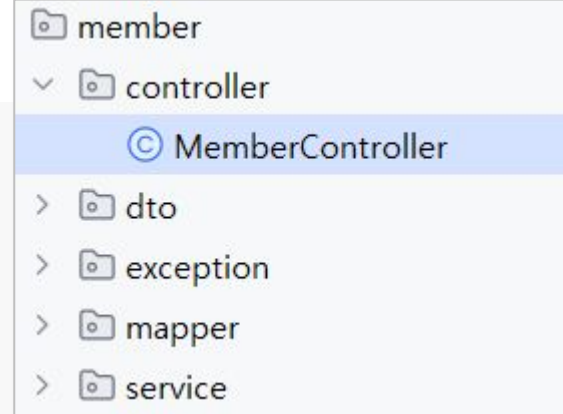
```
@PostMapping("")
```

```
public ResponseEntity<MemberDTO> join(MemberJoinDTO member) {
```

```
    return ResponseEntity.ok(service.join(member));
```

```
}
```

```
}
```



id중복체크

회원가입

- 컴포넌트 스캔을 설정

- **RootConfig.java**

```
@Configuration
@PropertySource({"classpath:/application.properties"})
@MapperScan(basePackages = {"org.scoula.board.mapper", "org.scoula.member.mapper"})
@ComponentScan(basePackages = {"org.scoula.board.service", "org.scoula.member.service"})
@EnableTransactionManagement
@Log4j2
public class RootConfig {
    ...
}
```

- **ServletConfig.java**

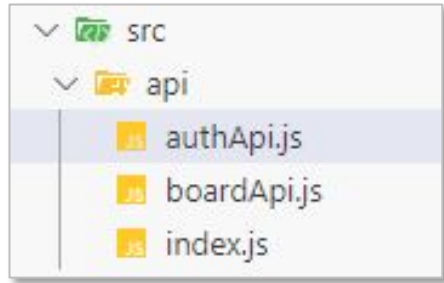
```
@EnableWebMvc
@ComponentScan(basePackages = {
    "org.scoula.controller",
    "org.scoula.exception",
    "org.scoula.board.controller",
    "org.scoula.member.controller",
})
public class ServletConfig implements WebMvcConfigurer {
```


2025년 상반기 K-디지털 트레이닝

회원관리 -회원가입 (프론트엔드)

[KB] IT's Your Life

- authApi 모듈



- **api/authApi.js**

```
import api from 'axios';

const BASE_URL = '/api/member';
const headers = { 'Content-Type': 'multipart/form-data' };

export default {
  // username 중복 체크, true: 중복(사용불가), false: 사용 가능
  async checkUsername(username) {
    const { data } = await api.get(`${BASE_URL}/checkusername/${username}`);
    console.log('AUTH GET CHECKUSERNAME', data);
    return data;
  },
}
```

● api/authApi.js

```
async create(member) {
```

```
// 아바타 파일 업로드 - multipart 인코딩 필요(FormData 객체 사용)
```

회원가입

```
const formData = new FormData();
formData.append('username', member.username);
formData.append('email', member.email);
formData.append('password', member.password);
```

```
if (member.avatar) {
  formData.append('avatar', member.avatar);
}
```

```
const { data } = await api.post(BASE_URL, formData, headers);
```

```
console.log('AUTH POST: ', data);
return data;
```

```
}
};
```

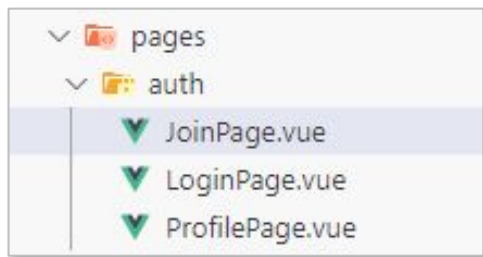
form객체 만들어 전달한 데이터 넣어줌.


아바타 첨부파일있으면 form객체에 넣어줌.

get방식이면 form만들어주지 않아도 됨.
첨부파일이 있는 경우 form객체에 넣어서 전달.
headers정보도 별도로 넣어주어야함.


비동기로 axios의 post함수
호출하여 form데이터
전달하고 결과 json으로
받아와 프린트


- 회원 가입 페이지







회원 가입


 사용자 ID : ID 중복 확인 이미 사용중인 ID입니다.

 아바타 이미지:

파일 선택 선택된 파일 없음

 email

 비밀번호:

 비밀번호 확인:

● router/auth.js

```
export default [  
  {  
    path: '/auth/join',  
    name: 'join',  
    component: () => import('../pages/auth/JoinPage.vue'),  
  },  
];
```

요청별 바인딩할 컴포넌트
지정

회원과 관련된
라우팅정보는 모두
auth.js에 정의

- **router/index.js**

```
import { createRouter, createWebHistory } from 'vue-router';
import HomePage from '../pages/HomePage.vue';
import authRouters from './auth';

const router = createRouter({
  history: createWebHistory(import.meta.env.BASE_URL),
  routes: [
    {
      path: '/',
      name: 'home',
      component: HomePage,
    },
    ...authRouters,
  ],
});

export default router;
```

• pages/auth/JoinPage.vue

```
<script setup>
import { reactive, ref } from 'vue';
import { useRouter } from 'vue-router';
import authApi from '@/api/authApi';
```

```
const router = useRouter();
```

```
const avatar = ref(null);
```

```
const checkError = ref('');
```

```
const member = reactive({ // 테스트용 초기화
  username: 'hong',
  email: 'hong@gmail.com',
  password: '12',
  password2: '12',
  avatar: null,
});
```

```
const disableSubmit = ref(true);
```

처음에 null(첨부파일 없음)

아바타 이미지:

파일 선택

선택된 파일 없음

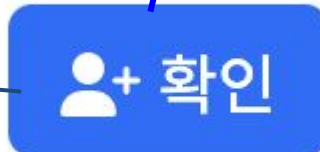
결과 메시지 출력

사용자 ID : ID 중복 확인 이미 사용중인 ID입니다.

hong

서버로
회원가입
시 전달될
값

id중복 확인을 누르지 않으면 확인버튼은 계속 비활성화 (disableSubmit = true)



처음에 비활성화 (눌러도 반응 없음) → 중복체크한 후, 활성화됨 (disableSubmit=false가 되어야 활성화됨)

• pages/auth/JoinPage.vue

// username 중복 체크

```
const checkUsername = async () => {
  if (!member.username) {
    return alert('사용자 ID를 입력하세요.');
```

```
  }
  disableSubmit.value = await authApi.checkUsername(member.username);
  console.log(disableSubmit.value, typeof disableSubmit.value);
```

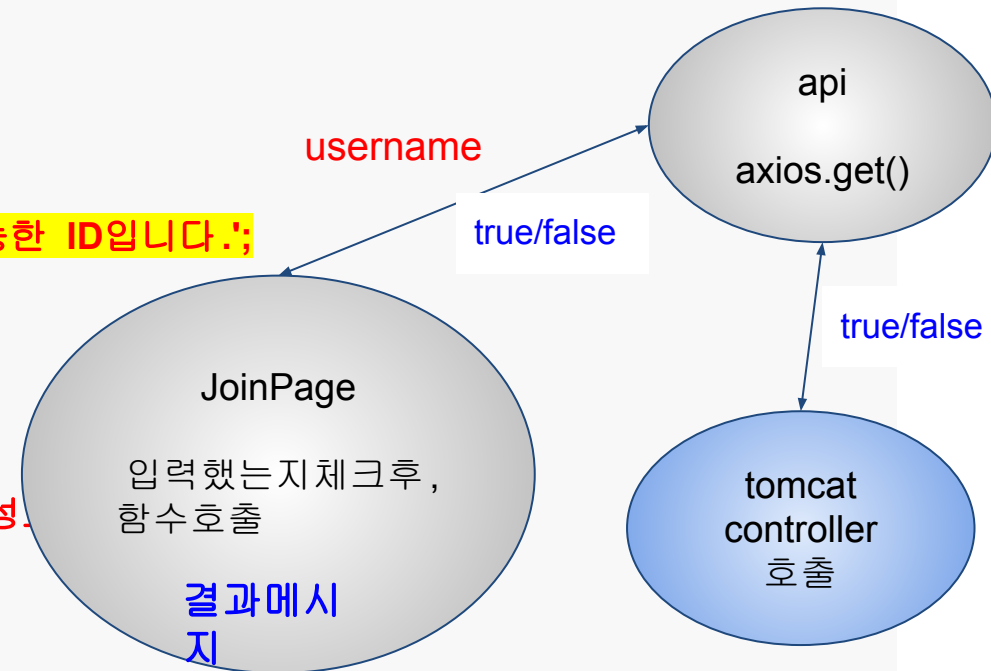
```
  checkError.value = disableSubmit.value ? '이미 사용중인 ID입니다.' : '사용가능한 ID입니다.';
};
```

// username 입력 핸들러

```
const changeUsername = () => {
  disableSubmit.value = true; //ID중복 체크를 하지 않았으므로 submit버튼 비활성
  if (member.username) {
    checkError.value = 'ID 중복 체크를 하셔야 합니다.';
  } else {
    checkError.value = "";
  }
};
```

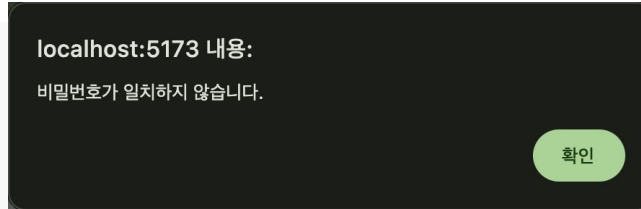
사용자 ID : ID 중복 확인 이미 사용중인 ID입니다.

hong



• pages/auth/JoinPage.vue

```
const join = async () => {
  if (member.password !== member.password2) {
    return alert('비밀번호가 일치하지 않습니다.');
```



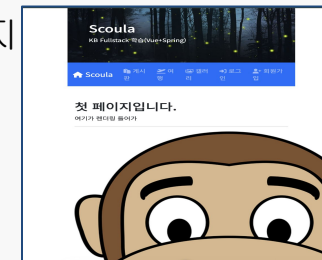
```
if (avatar.value.files.length > 0) {
  member.avatar = avatar.value.files[0];
}
```

👤 아바타 이미지:

파일 선택

선택된 파일 없음

```
try {
  await authApi.create(member); // 회원가입
  router.push({ name: 'home' }); // 회원 가입 성공 시, 첫 페이지로 이동 또는 로그인 페이지
} catch (e) {
  console.error(e);
}
};
</script>
```



• pages/auth/JoinPage.vue

```
<template>
```

```
<div class="mt-5 mx-auto" style="width: 500px">
```

```
<h1 class="my-5">
```

```
<i class="fa-solid fa-user-plus"></i>
```

회원 가입

```
</h1>
```

form내 submit버튼을 눌렀을 때 바로 전송하는 기본 기능을 막고, join함수를 호출

```
<form @submit.prevent="join">
```

```
<div class="mb-3 mt-3">
```

```
<label for="username" class="form-label">
```

```
<i class="fa-solid fa-user"></i>
```

사용자 ID :

클릭하면 중복체크함수 호출
입력여부확인 → axios.get(username) 중복체크 호출

```
<button type="button" class="btn btn-success btn-sm py-0 me-2" @click="checkUsername">
```

ID 중복 확인</button>

disableSubmit.value

- true이면 에러메시지 blue
- false이면 에러메시지 red

```
<span :class="disableSubmit.value ? 'text-primary' : 'text-danger'">{{ checkError }}</span>
```

```
</label>
```

```
<input type="text" class="form-control" placeholder="사용자 ID" id="username"
```

```
@input="changeUsername" v-model="member.username" />
```

```
</div>
```

input태그에 입력하면 input의 value에도 들어가고 member.username에도 들어가고, 반대도 가능(양방향 바인딩 cf)v-bind)

● pages/auth/JoinPage.vue

```
<div>
  <label for="avatar" class="form-label">
    <i class="fa-solid fa-user-astronaut"></i>
    아바타 이미지:
  </label>
  <input type="file" class="form-control" ref="avatar" id="avatar" accept="image/png, image/jpeg" />
</div>
```

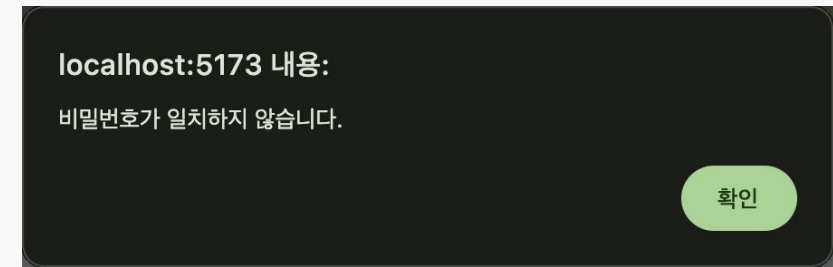
파일첨부는 type="file"이라고
해야함

파일첨부를 하면 동시에 avatar값 변경이 인식되어 avatar변수에
들어감.
**const avatar = "첨부파일"; → 서버로 전달할 값의 키:값으로
들어감.**

```
<div class="mb-3 mt-3">
  <label for="email" class="form-label">
    <i class="fa-solid fa-envelope"></i>
    email
  </label>
  <input type="email" class="form-control" placeholder="Email" id="email" v-model="member.email" />
</div>
```

• pages/auth/JoinPage.vue

```
<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" id="password" v-model="member.password" />
</div>
<div class="mb-3">
  <label for="password" class="form-label">
    <i class="fa-solid fa-lock"></i> 비밀번호 확인:
  </label>
  <input type="password" class="form-control" placeholder="비밀번호 확인" id="password2" v-model="member.password2" />
</div>
<button type="submit" class="btn btn-primary mt-4" :disabled="disableSubmit">
  <i class="fa-solid fa-user-plus"></i>
  확인
</button>
</form>
</div>
```



처음에 비활성화 (눌러도 반응 없음) →
중복체크한 후, 활성화됨
(disableSubmit=false가 되어야 활성화됨)

```
</template>
```

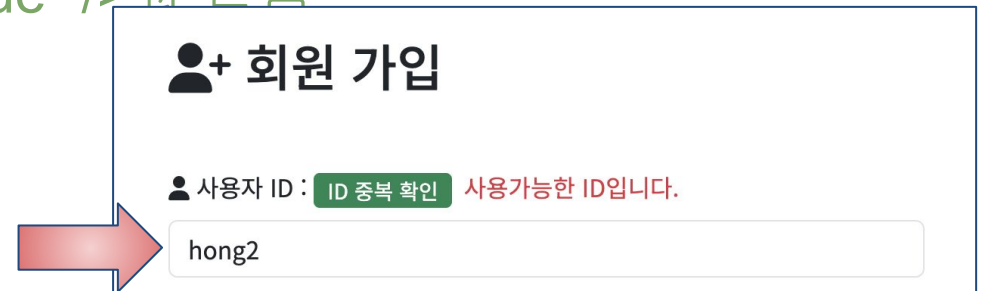

- input(글자를 한글자씩 칠 때마다 changeUsername 함수 호출함.)
 - v-mode이므로 input의 value도 입력한 값으로 바뀌고, member.username값도 함께 바뀜
 - member.username 값을 input 요소와 양방향 바인딩함.
 - 즉, 사용자가 입력창에 값을 입력하면 member.username의 값도 자동으로 바뀌고, 반대로 member.username이 변경되면 입력창의 값도 바뀜.
 - <input :value="member.username"
- @input="member.username = \$event.target.value" />과 같음



👤+ 회원 가입

👤 사용자 ID : ID 중복 확인 이미 사용중인 ID입니다.

hong



👤+ 회원 가입

👤 사용자 ID : ID 중복 확인 사용가능한 ID입니다.

hong2

AUTH POST:

```
{username: 'hong2', email: 'hong@gmail.com', regDate: 1722842581000, updateDate: 1722842581000, authList: Array(1)}  
  ▶ authList: ['ROLE_MEMBER']  
    email: "hong@gmail.com"  
    regDate: 1722842581000  
    updateDate: 1722842581000  
    username: "hong2"  
  ▶ [[Prototype]]: Object
```

authApi.js:36

첫 페이지로 redirect되면서
응답받음.

