KB금융그룹 | 국민의 평생 금융파트너

FE에서는 axios를 통해서 api통신해야하고 pinia를 통해서 상태관리를 해야한다
pinia 상태관리 에 따라(로그인 했냐 안했냐) 화면을 달리 구성하게끔 하자

2025년 상반기 K-디지털 트레이닝

회원관리-회원가입(백엔드)

[KB] IT's Your Life

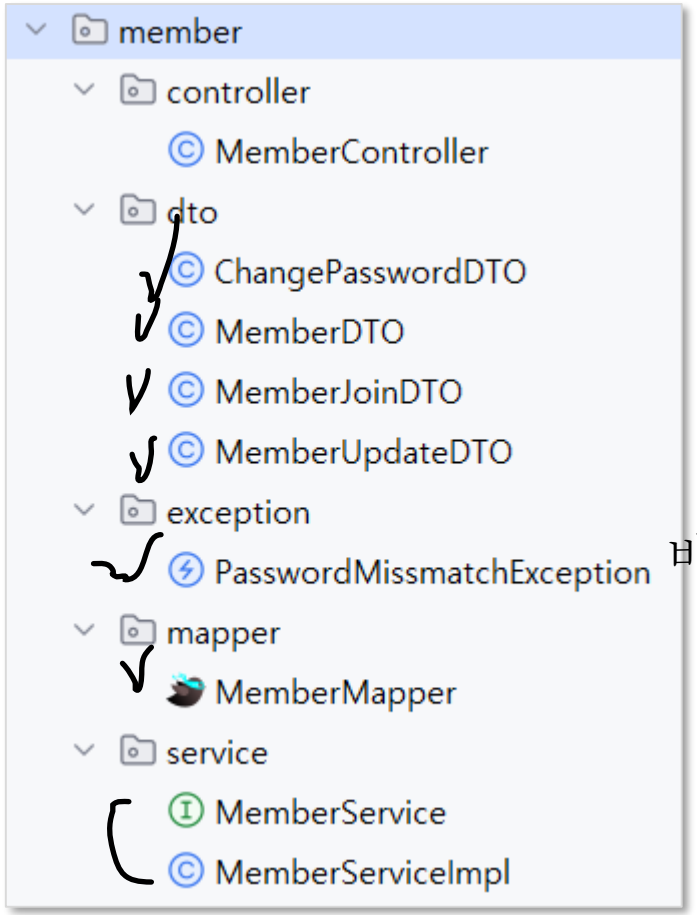
BE에서는 DB구축하고 jwt관리.

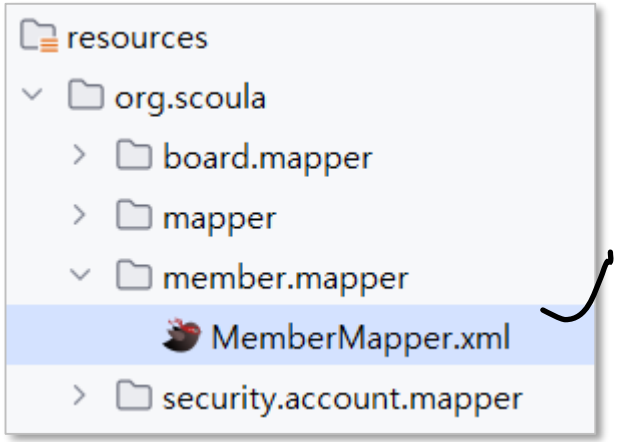
KB국민은행

백엔드 준비하기

회원관리 기본 패키지

- org.scoula.member





비번 불일치시 예외 처리

회원관리 – 회원 가입

회원 가입

회원 가입시 고려사항

- username 중복 여부 점검
- 비밀번호 암호화 — password encoder 주입받아서
- 회원 정보 저장시 회원 권한도 같이 저장 → 트랜잭션 처리 — 두번의 insert
- 회원 아바타 이미지 파일 업로드 → c:/upload/avatar/<username>.jpg 형태로 저장
  - multipart 인코딩을 해야한다/ json 인코딩이 아닌.
  - multipart 인코딩은 security에 영향을 미쳐서 조정해야한다.
  - 계정당 이미지 1대1 관계로. 1대n관계라면 테이블 또 만들어야함.
  - 파일명 규칙도 정하고. 1대1이면 그냥 저장.

DTO

- MemberJoinDTO
- MemberDTO

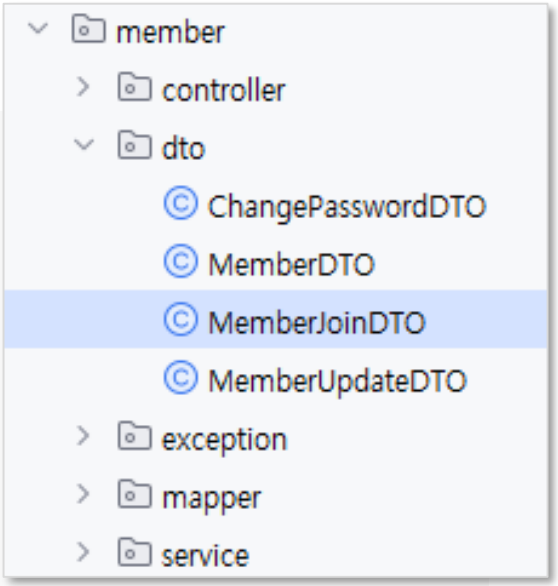
회원관리 – 회원 가입

MemberJoinDTO.java

디비 테이블과 일치 | 시켜 놓은

```
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;
```

회원가입할때 정보 담는 객체



회원관리 – 회원 가입

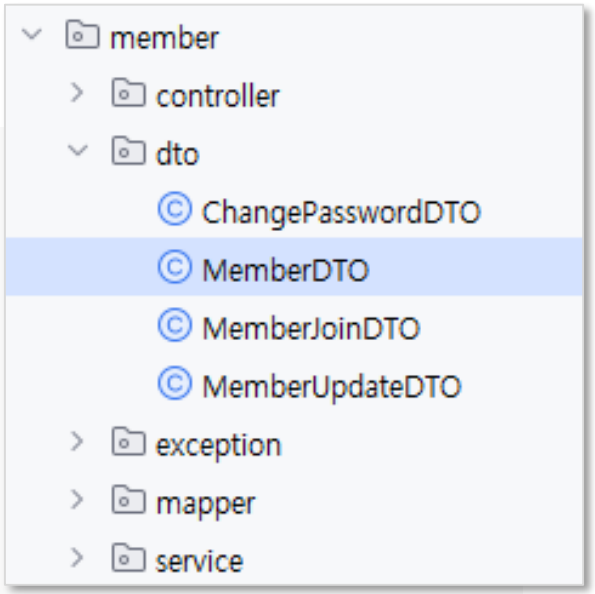
MemberJoinDTO.java

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

}
```

회원 가입정보 => 회운정보

- 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
```

회원정보를 활용할때 사용할 객체

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();
    }
}
```

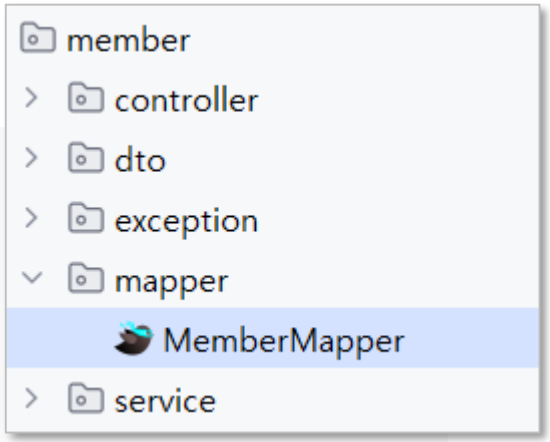
멤버 VO => 멤버 DTO

멤버DTO => 멤버VO

회원관리 – 회원 가입

MemberMapper.java

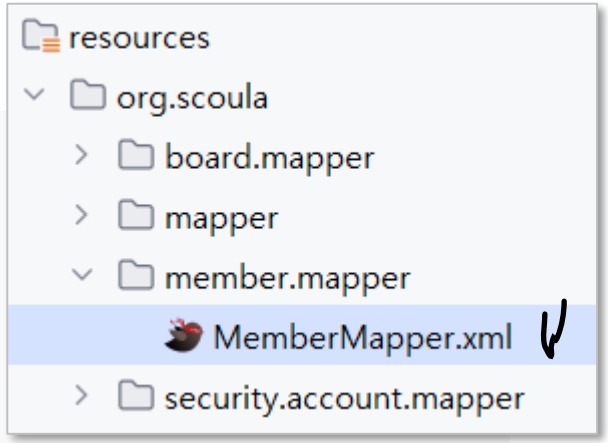
멤버정보에 대한 매퍼 정의




```
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); //이건 조인을 통해서 전체 정보(권한목록까지)를 같이 가져오는것

    MemberVO findByUsername(String username); // id 중복 체크시 사용V 조인 없이 사용자 명만 가져오는거 시도

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

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

}
```

MemberMapper.xml

매퍼 xml 정의




```
<?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>
```

1대N

1대1은 association인거 기억하죠?

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>
```

view를 만들어서 활용해도 좋아

회원관리 – 회원 가입

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>
```

멤버 테이블에 대한 insert

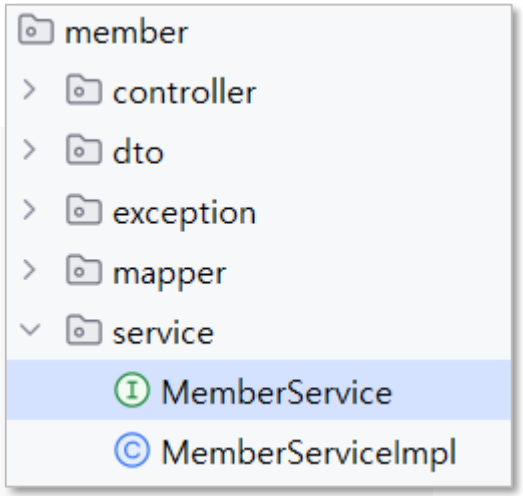
이때 들어가는 password는 이미 앞에서 암호화가 되어있어야 함.

권한 테이블에 대한 insert

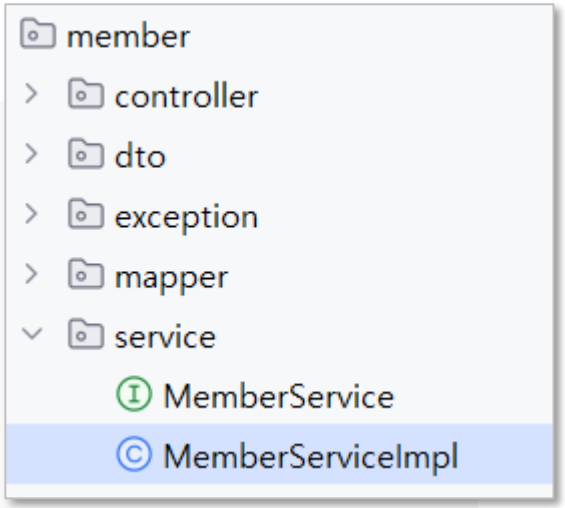
MemberService.java

```
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);
}
```

매퍼를 이용해서 디비와 연동을 지시할 서비스 정의



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;
    }
```

사용할 암호화 인코더와 매퍼 주입받기

중복체크

찾으면 이미 있는 이름이니까 true

MemberServiceImpl.java

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

private void saveAvatar(MultipartFile avatar, String username) {
    //아바타 업로드
    if(avatar != null && !avatar.isEmpty()) {
        File dest = new File("c:/upload/avatar", username + ".png");
        try {  //이미지 조정 작업이 들어갈 수 있다. 타입 변환, 크기변환. thumnail라이브러리 사용하면 쉬워
            avatar.transferTo(dest);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}
```

예외가 없다면 변환하고 반환.

@Transactional에 반응하게
런타임 예외로 변환해서 던짐.

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());
    }

}
```

회원가입

아까 말한 두번의 insert

insert 두번
이미지 저장

에서 예외발생 가능

회원관리 – 회원 가입

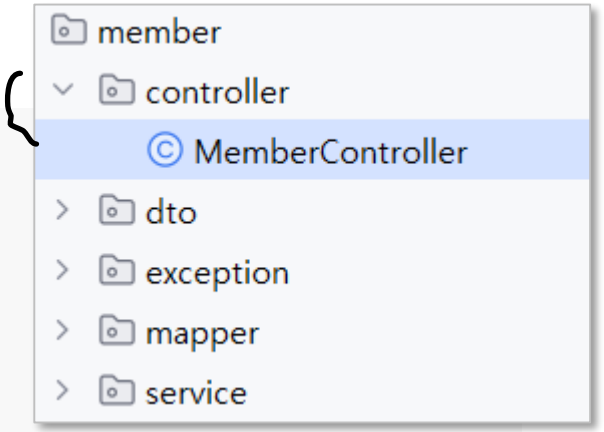
컨트롤러

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

회원관리 – 회원 가입

MemberController.java

ResponseEntity.ok( ).body( )
ResponseEntity.build( )




```
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));
    }
}
```

경로변수

@PathVariable밸류값 생략시
주의 경로변수명과 매개변수명 같아야함

JSON형식.
바디 내용은 불린 값.

회원가입한 결과 : 멤버DTO

회원관리 – 회원 가입

- **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 {
    …
}
```

새로운 매퍼와 서비스 스캔되게

ServletConfig.java

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

새로운 컨트롤러 스캔되게

KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

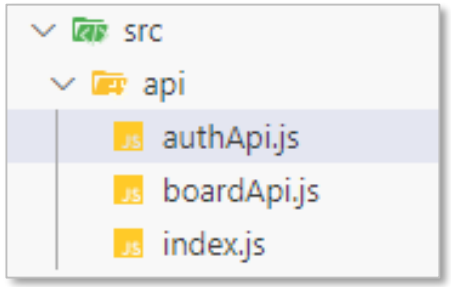
회원관리-회원가입(프론트엔드)

[KB] IT's Your Life

KB 국민은행

회원관리 – 회원 가입

authApi 모듈



기능별로 api모듈 만들어.
각각의 서비스별로 필요한 걸 정의.

회원관리 – 회원 가입

api/authApi.js 인증과 관련된 api

```
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;
  },
```

객체 export하고 있음

만들어진 url로 api get요청 //비동기 동작

역직렬화된 js객체

security config에서 configure메서드에서
web.ignoring( ).antMatchers(... "/api/member/**" ...)

회원관리 – 회원 가입

api/authApi.js

```
  async create(member) {
    // 아바타 파일 업로드 – multipart 인코딩 필요 → FormData 객체 사용

    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;
  }
};
```

//html의form을 캡슐화한 js객체. form에 대한 설정을 할 수 있다

<input name="?" value="?" ...>

formData.append('네임값', '밸류값')

유저가 아바타 등록했다면

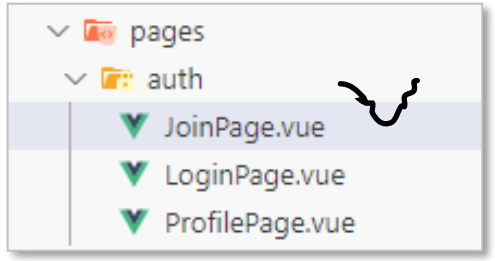
<input type="file" name="avatar" ...>

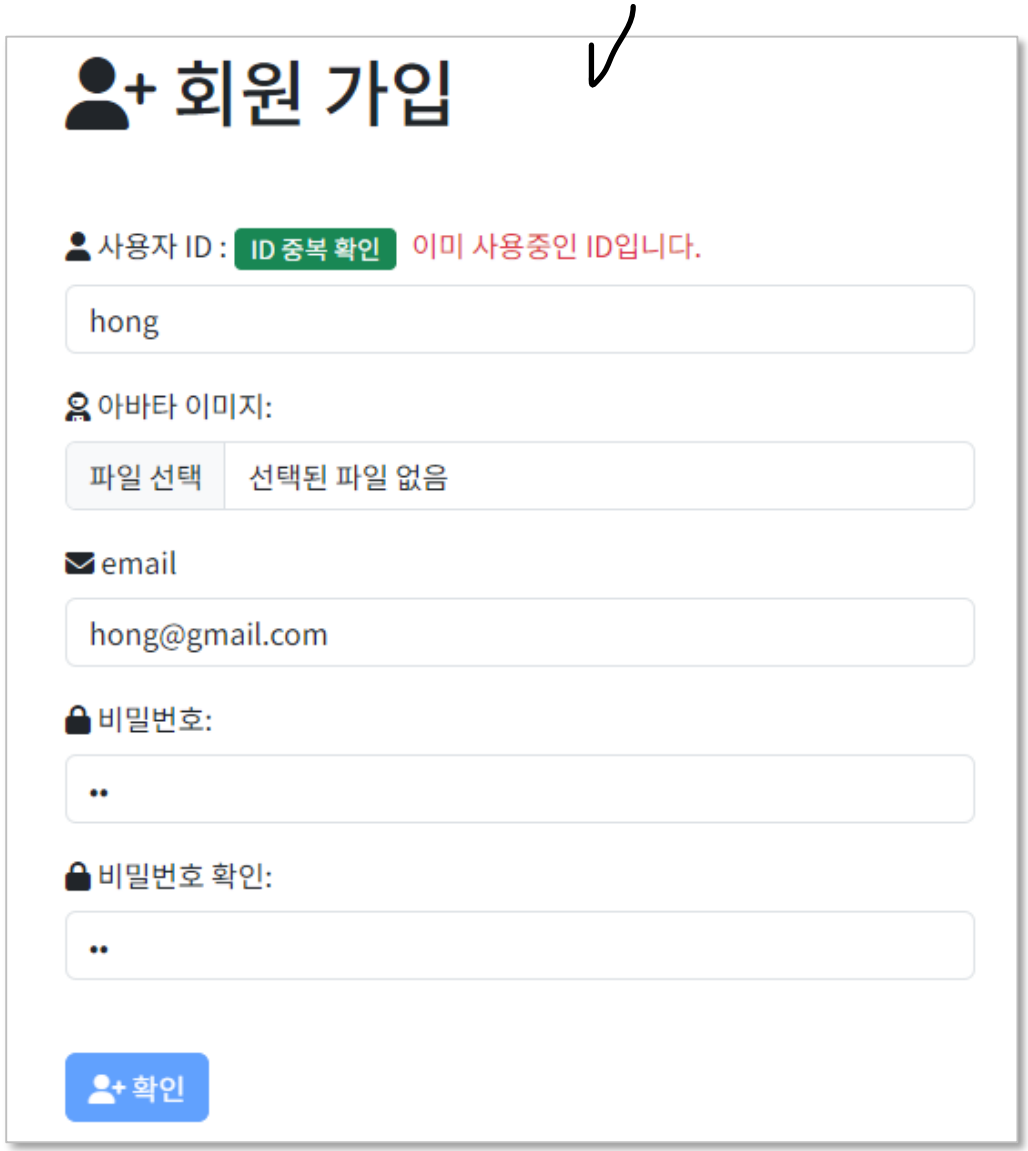
요청 url과 폼데이터와 헤더
post요청 보내기

응답으로 멤버DTO의 내용이 JSON으로

회원관리 – 회원 가입

회원 가입 페이지





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);
```

초기값이 null? 단일값?

submit버튼 비활성화 여부 변수

회원관리 – 회원 가입

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;
  if (member.username) {
    checkError.value = 'ID 중복 체크를 하셔야 합니다.';
  } else {
    checkError.value = '';
  }
};
```

중복 확인 버튼의 이벤트 핸들러로 쓰임

입력할때 바로 호출되어야함. 유저의 이름에 변화가 생기면.

중복 체크 누르면 중복체크 이벤트 핸들러가 동작하고 submit버튼 활성/비활성

회원관리 – 회원 가입

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]; 파일처리
  }

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

    <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 :
          <button type="button" class="btn btn-success btn-sm py-0 me-2" @click="checkUsername">ID 중복 확인</button>
          <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>
```

default submit이벤트 방지 및 핸들러 지정

버튼 클릭 핸들러 지정

판단 변수 활용

입력할때마다 호출되는 핸들러 지정

변수와 양방향 바인딩

회원관리 – 회원 가입

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>

<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>
```

얘는 반대. 여기서 js에게 배정

템플릿에서 만든거를 js에서 사용. 그래서 앞에서 avatar=ref(null)한거임

초기화 과정이 서로 반대

js에서 초기화된거 사용

js 프록시 변수 지정

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>
</template>
```

회원관리 – 회원 가입


AUTH POST:
authApi.js:36
{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