

2025년 상반기 K-디지털 트레이닝

회원관리-회원가입(백엔드)

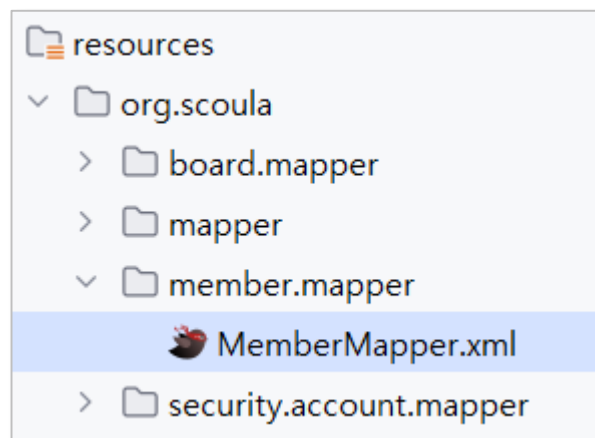
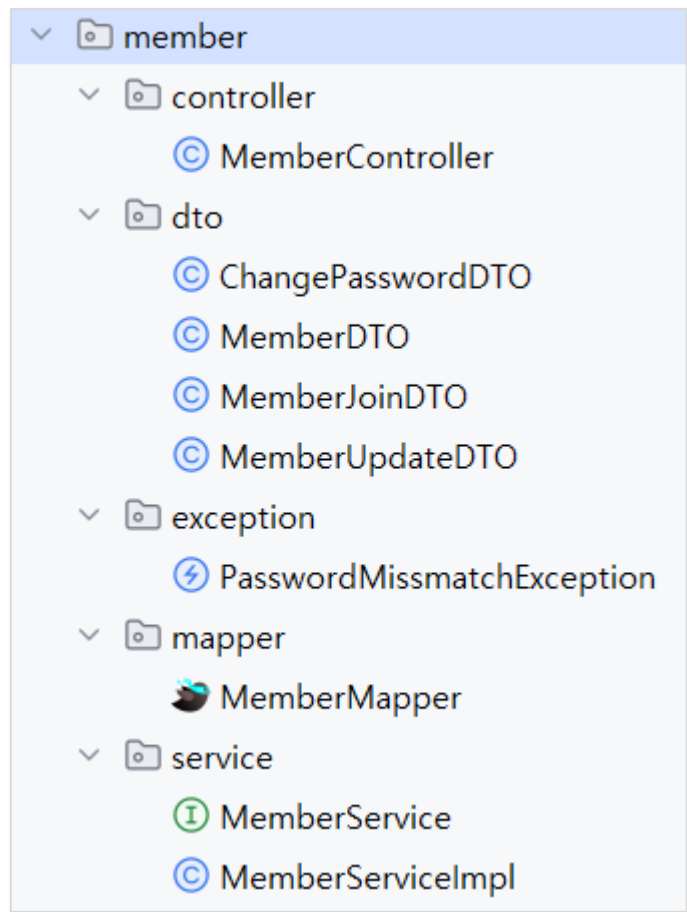
[KB] IT's Your Life

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
- MemberDTO

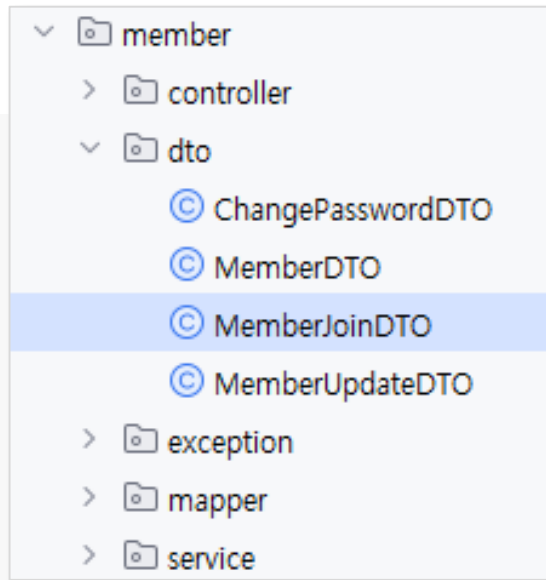
MemberJoinDTO.java

```
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;

...

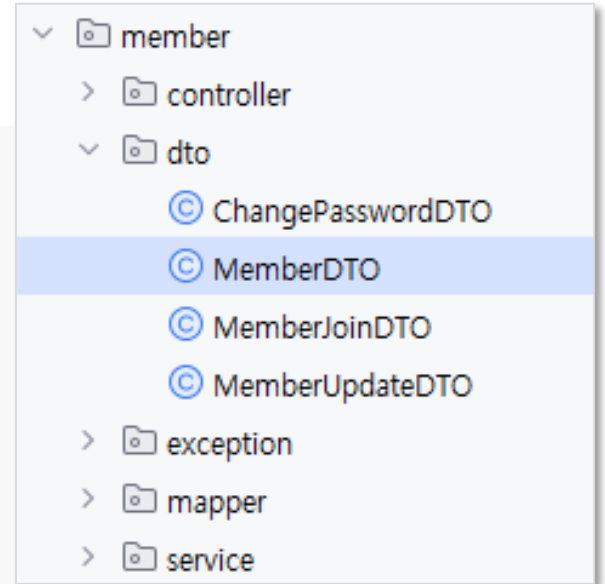
import org.scoula.security.account.domain.MemberVO;
import org.springframework.web.multipart.MultipartFile;

...

@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;

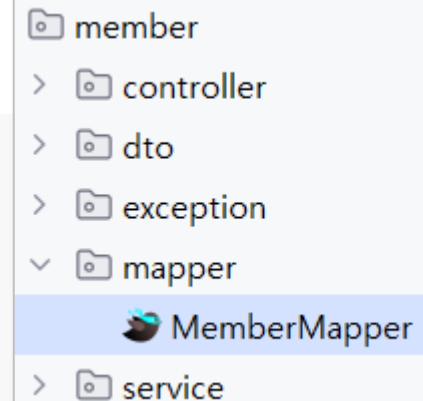
...
import org.scoula.security.account.domain.AuthVO;
import org.scoula.security.account.domain.MemberVO;

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
}
```

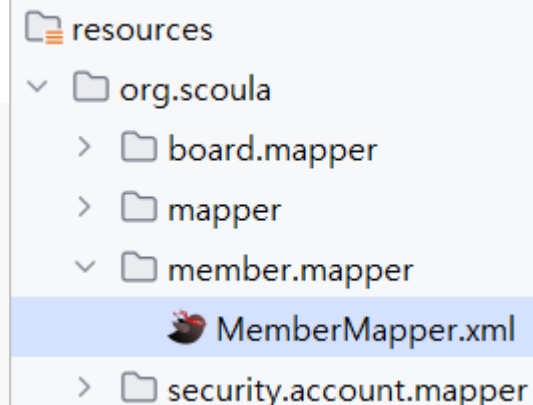


MemberMapper.xml

```
<?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

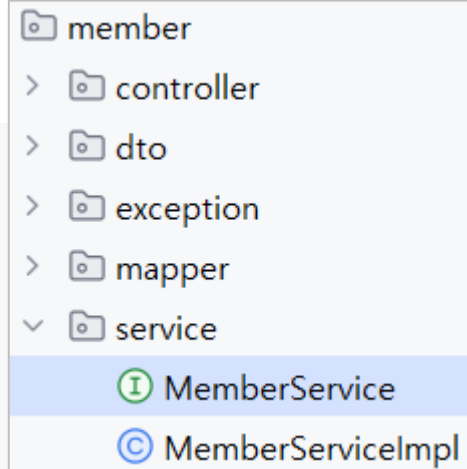
```
package org.scoula.member.service;

import org.scoula.member.dto.MemberJoinDTO;

public interface MemberService {
    boolean checkDuplicate(String username);

    MemberDTO get(String username);

    MemberDTO join(MemberJoinDTO member);
}
```

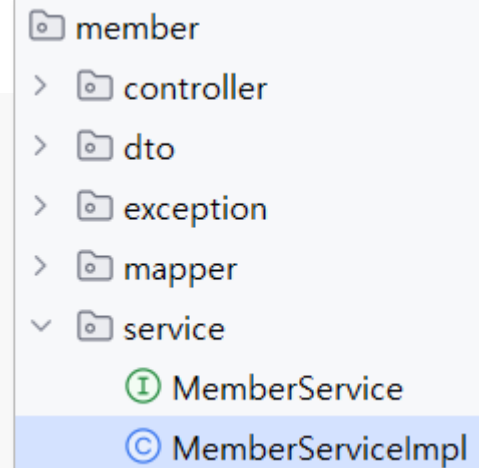


MemberServiceImpl.java

```
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;
    }
}
```



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 {
            avatar.transferTo(dest);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}
```

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());
}
}
```

✓ 컨트롤러

○ URL

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

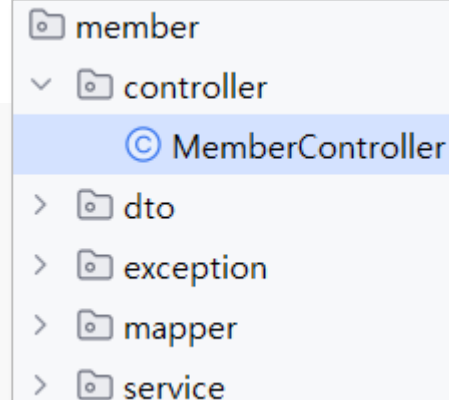
MemberController.java

```
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));
    }
}
```



- ✓ 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

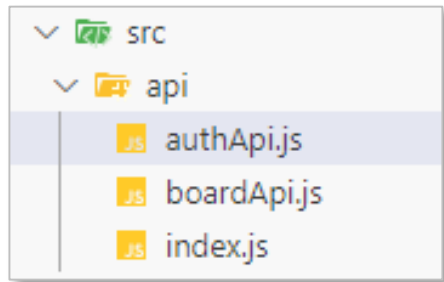
```
@EnableWebMvc
@ComponentScan(basePackages = {
    "org.scoula.controller",
    "org.scoula.exception",
    "org.scoula.board.controller",
    "org.scoula.member.controller",
})
public class ServletConfig implements WebMvcConfigurer {
```

2025년 상반기 K-디지털 트레이닝

회원관리-회원가입(프론트엔드)

[KB] IT's Your Life

✓ authApi 모듈



api/authApi.js

```
import api from 'axios';

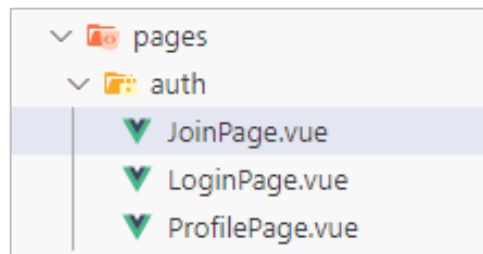
const BASE_URL = '/api/member';
const headers = { 'Content-Type': 'multipart/form-data' };

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  
  
  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;  
}  
};
```


✓ 회원 가입 페이지



+ 회원 가입

 사용자 ID :

ID 중복 확인 이미 사용중인 ID입니다.


 아바타 이미지:


파일 선택

선택된 파일 없음

 email

 비밀번호:

 비밀번호 확인:

+ 확인

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);
```

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 = '';
  }
};
```

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];  
  }  
  
  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);  
  }  
};  
</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>
    </form>
  </div>
```

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>
```

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>
```

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
```