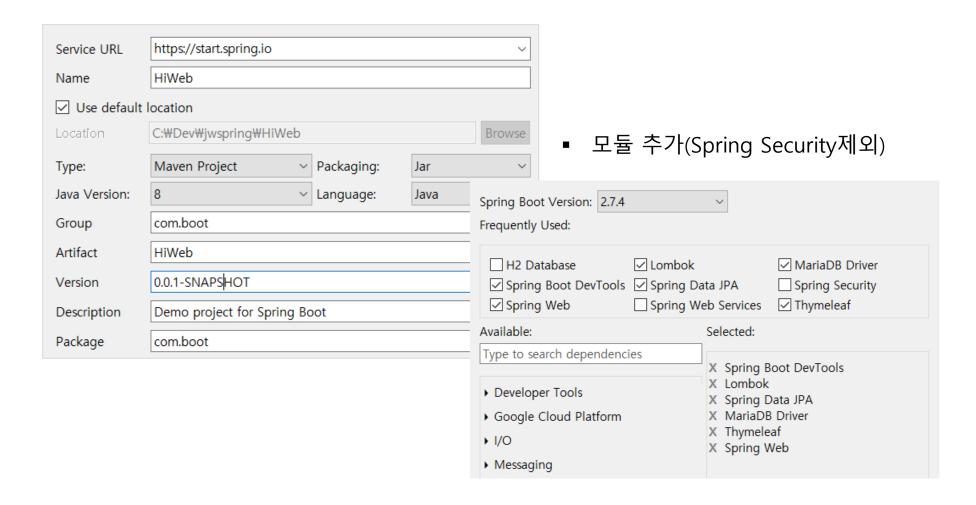
### 8장. 웹 애플리케이션 통합

Hiweb



#### ■ 프로젝트 생성 및 설정

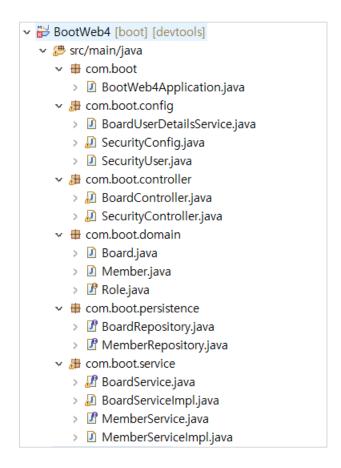


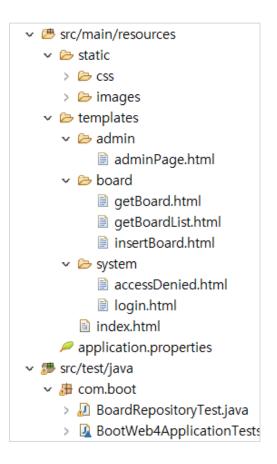
```
# WebApplication Type Setting
# spring.main.web-application-type=none
spring.main.web-application-type=servlet
# DataSource Setting
spring.datasource.driver-class-name=org.mariadb.jdbc.Driver
spring.datasource.url=jdbc:mariadb://127.0.0.1:3306/hiweb
spring.datasource.username=root
spring.datasource.password=12345
# JPA Setting
spring.jpa.hibernate.ddl-auto=update
spring.jpa.generate-ddl=false
spring.jpa.show-sql=true
spring.jpa.database-platform=org.hibernate.dialect.MariaDB103Dialect
spring.jpa.properties.hibernate.format_sql=true
# static resource
spring.mvc.static-path-pattern=/static/**
# Security log level Setting
logging.level.org.springframework.web=debug
logging.level.org.springframework.security=debug
```

■ MariaDB - 데이터베이스(hiweb)



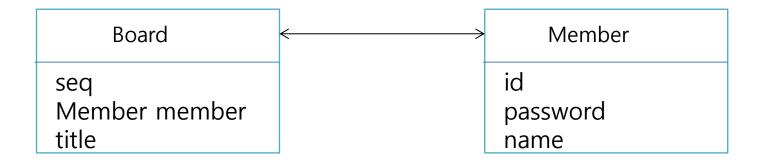
#### ■ 전체 프로젝트 계층 구조도





# 연관 관계 매핑

● 양방향 매핑 설정하기



회원과 게시판 엔티티가 양방향으로 서로 참조할 수 있도록 연관 매핑을 추가한다. 게시판과 회원은 다대일(N:1) 양방향 관계다.

게시판은 자신을 등록한 회원을 Board.member 변수로 참조하며, 회원은 자신이 등록한 게시글 목록을 Member.boardList 컬렉션으로 참조한다.

■ 엔티티 클래스 만들기

```
@ToString(exclude="member")
@Getter
@Setter
@Entity
public class Board implements Serializable{
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue
    private Long seq;
    private String title;
    private String content;
    @Column(updatable=false,
            columnDefinition = "timestamp DEFAULT CURRENT TIMESTAMP")
    private Date createDate;
```

■ 엔티티 클래스 만들기

#### ■ 엔티티 클래스 만들기

```
@ToString(exclude="boardList")
@Getter
@Setter
@Entity
public class Member implements Serializable{
    private static final long serialVersionUID = 1L;
    @Id
    @Column(name="MEMBER ID")
    private String id;
    private String password;
    private String name;
    @Enumerated(EnumType.STRING)
    private Role role;
    private boolean enabled;
    @OneToMany(mappedBy="member", fetch=FetchType.EAGER)
    private List<Board> boardList = new ArrayList<>();
```

■ 리포지터리 인터페이스 만들기

```
public interface BoardRepository extends JpaRepository<Board, Long>{
    //글 목록 검색
    @Query("SELECT b FROM Board b")
    Page<Board> getBoardList(Pageable pageable);

    //조회수 증가
    @Modifying
    @Query("UPDATE Board b SET b.cnt = b.cnt + 1 WHERE b.seq = :seq")
    void updateCount(Long seq);
}
```

```
public interface MemberRepository extends JpaRepository<Member, String>{
}
```

#### ■ JPA 테스트하기

```
@SpringBootTest
public class BoardRepositoryTest {
    @Autowired
    private MemberRepository memberRepo;
    @Autowired
    private BoardRepository boardRepo;
    @Test
    public void testInsert() {
        //회원 입력
        Member member1 = new Member();
        member1.setId("member");
        member1.setPassword("member123");
        member1.setName("회원");
        member1.setRole(Role.ROLE_MEMBER);
        member1.setEnabled(true);
        memberRepo.save(member1);
```

```
//관리자 입력
Member member2 = new Member();
member2.setId("admin");
member2.setPassword("admin123");
member2.setName("관리자");
member2.setRole(Role.ROLE_ADMIN);
member2.setEnabled(true);
memberRepo.save(member2);
//회원이 작성한 게시글 입력
for(int i = 1; i <= 13; i++) {
    Board board = new Board();
    board.setMember(member1);
    board.setTitle(member1.getName() + "이 등록한 게시글" + i);
    board.setContent(member1.getName() + "가 등록한 게시글" + i);
    boardRepo.save(board);
//관리자가 작성한 게시글 입력
for(int i = 1; i \le 3; i++) {
    Board board = new Board();
    board.setMember(member2);
    board.setTitle(member2.getName() + "이 등록한 게시글" + i);
    board.setContent(member2.getName() + "가 등록한 게시글" + i);
    boardRepo.save(board);
```

admin 1 관리자 admin123 ROLE_ADMIN member 1 회원 member123 ROLE_MEMBER	member_id	7	enabled	name	password	role
member 1 회원 member123 ROLE_MEMBER	admin		1	관리자	admin123	ROLE_ADMIN
	member		1	회원	member123	ROLE_MEMBER

seq 🢡	cnt	content	create_date	title	member_id 💡
1	0	김기용가 등록한 게시글1	2022-09-29 05:34:02	김기용이 등록한 게시글1	member
2	0	김기용가 등록한 게시글2	2022-09-29 05:34:02	김기용이 등록한 게시글2	member
3	0	김기용가 등록한 게시글3	2022-09-29 05:34:02	김기용이 등록한 게시글3	member
4	0	김기용가 등록한 게시글4	2022-09-29 05:34:02	김기용이 등록한 게시글4	member
5	0	김기용가 등록한 게시글5	2022-09-29 05:34:02	김기용이 등록한 게시글5	member
6	0	김기용가 등록한 게시글6	2022-09-29 05:34:02	김기용이 등록한 게시글6	member
7	0	김기용가 등록한 게시글7	2022-09-29 05:34:02	김기용이 등록한 게시글7	member
8	0	김기용가 등록한 게시글8	2022-09-29 05:34:02	김기용이 등록한 게시글8	member
9	0	김기용가 등록한 게시글9	2022-09-29 05:34:02	김기용이 등록한 게시글9	member
10	0	김기용가 등록한 게시글10	2022-09-29 05:34:02	김기용이 등록한 게시글10	member
11	0	김기용가 등록한 게시글11	2022-09-29 05:34:02	김기용이 등록한 게시글11	member
12	0	김기용가 등록한 게시글12	2022-09-29 05:34:02	김기용이 등록한 게시글12	member
13	0	김기용가 등록한 게시글13	2022-09-29 05:34:02	김기용이 등록한 게시글13	member
14	0	관리자가 등록한 게시글1	2022-09-29 05:34:02	관리자이 등록한 게시글1	admin
15	0	관리자가 등록한 게시글2	2022-09-29 05:34:02	관리자이 등록한 게시글2	admin
16	0	관리자가 등록한 게시글3	2022-09-29 05:34:02	관리자이 등록한 게시글3	admin

#### ■ 상세 조회 테스트

```
//회원과 회원이 작성한 게시글 상세 정보
@Test

public void testGetBoard() {
    Board board = boardRepo.findById(1L).get();

    System.out.println("[" + board.getSeq() + "번 게시글 정보 ]");
    System.out.println("제목\t: " + board.getTitle());
    System.out.println("작성자\t: " + board.getMember().getName());
    System.out.println("내용\t: " + board.getContent());
    System.out.println("등록일\t: " + board.getCreateDate());
    System.out.println("조회수\t: " + board.getCnt());
}
```

```
[1번 게시글 정보 ]제목 : 김기용이 등록한 게시글1작성자 : 김기용내용 : 김기용가 등록한 게시글1등록일 : 2022-09-29 05:34:02.0조회수 : 0
```

#### ■ 목록 검색 테스트

```
//회원(member) 또는 관리자(admin)가 등록한 게시글 목록 검색
@Test
public void testGetBoardList() {
    Member member = memberRepo.findById("member").get();
    //Member member = memberRepo.findById("admin").get();

System.out.println("[ " + member.getName() + "이 등록한 게시글 ]");
    for(Board board : member.getBoardList()) {
        System.out.println("--->" + board.toString());
    }
}
```

■ 서비스 인터페이스와 클래스 만들기

```
public interface BoardService {
   //List<Board> getBoardList(); //게시글 목록
   Page < Board > getBoardList(Board board); //게시글 목록(페이징 처리)
   void insertBoard(Board board); //게시글 등록
   Board getBoard(Long seq); //게시글 상세보기
   void updateBoard(Board board); //게시글 수정
   void deleteBoard(Board board); //게시글 삭제
   void updateCount(Long seq); //조회수
```

■ 서비스 인터페이스와 클래스 만들기

```
@Service
public class BoardServiceImpl implements BoardService {
    @Autowired
    private BoardRepository boardRepo;
    //게시글 목록 보기
   @Override
    public Page<Board> getBoardList(Board board) {
       //10개의 데이터를 내림차순 정렬(페이지 번호, 검색 데이터 개수, 내림차순)
       Pageable pageable = PageRequest.of(0, 10, Sort.Direction.DESC, "seq");
       return boardRepo.getBoardList(pageable);
    //게시글 등록
    @Override
    public void insertBoard(Board board) {
       boardRepo.save(board);
```

```
//게시글 상세 조회
@Override
public Board getBoard(Long seq) {
    return boardRepo.findById(seq).get();
//게시글 수정
@Override
public void updateBoard(Board board) {
    boardRepo.save(board);
//게시글 삭제
@Override
public void deleteBoard(Board board) {
    boardRepo.delete(board);
//조회수
@Transactional
@Override
public void updateCount(Long seq) {
    boardRepo.updateCount(seq);
```

게시판 프로그램입니다.



글 목록 로그인

■ 인덱스 페이지 추가 – index.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="utf-8">
<title>메인 페이지</title>
<link rel="stylesheet" href="/static/css/style.css">
</head>
<body>
    <div id="container">
       <h2>게시판 프로그램입니다.</h2>
       <img src="/static/images/healing.jpg" alt="폭포 사진">
       <a th:href="@{board/getBoardList}">글 목록</a>&nbsp;&nbsp;&nbsp;
          <a th:href="@{system/Login}">로그인</a>
       </div>
</body>
</html>
```

■ 컨트롤러 만들기 - BoardController

```
@RequestMapping("/board/")
@Controller
public class BoardController {
    @Autowired
    private BoardService service;
    //게시글 목록
    @GetMapping("/getBoardList")
    public String getBoardList(Board board, Model model) {
        Page<Board> boardList = service.getBoardList(board);
        model.addAttribute("boardList", boardList);
        return "board/getBoardList";
    //게시글 등록 폼 요청
    @GetMapping("/insertBoard")
    public String insertBoardView() {
        return "board/insertBoard";
```

```
//게시글 등록 처리
@PostMapping("/insertBoard")
public String insertBoard(Board board,
       @AuthenticationPrincipal SecurityUser principal) {
   //로그인 성공한 Member(회원) 객체를 가지고 있는 SecurityUser 객체를
   //매개 변수로 받음. 연관된 Member 엔티티를 Board 엔티티에 설정함
    board.setMember(principal.getMember());
    service.insertBoard(board);
    return "redirect:getBoardList";
//게시글 상세 조회
@GetMapping("/getBoard")
public String getBoard(Long seq, Model model) {
    service.updateCount(seq); //조회수 증가
    Board board = service.getBoard(seq);
    model.addAttribute(board);
    return "board/getBoard";
//게시글 삭제
@GetMapping("/deleteBoard")
public String deleteBoard(Board board) {
    service.deleteBoard(board);
    return "redirect:getBoardList";
```

#### 게시글 목록

관리자 님 환영합니다.....로그아웃 게시판 관리

번호	제목	작성자	등록일	조회수
16	관리자이 등록한 게시글3	관리자	2022-09-29 05:34:02	0
15	관리자이 등록한 게시글2	관리자	2022-09-29 05:34:02	0
14	관리자이 등록한 게시글1	관리자	2022-09-29 05:34:02	0
13	김기용이 등록한 게시글13	김기용	2022-09-29 05:34:02	0
12	김기용이 등록한 게시글12	김기용	2022-09-29 05:34:02	0
11	김기용이 등록한 게시글11	김기용	2022-09-29 05:34:02	0
10	김기용이 등록한 게시글10	김기용	2022-09-29 05:34:02	0
9	김기용이 등록한 게시글9	김기용	2022-09-29 05:34:02	0
8	김기용이 등록한 게시글8	김기용	2022-09-29 05:34:02	0
7	김기용이 등록한 게시글7	김기용	2022-09-29 05:34:02	0

새글 등록

■ 글 목록 화면 작성 – getBoardList.html

```
<html xmlns:th="http://www.thymeleaf.org"</pre>
     xmlns:sec="http://www.thymeleaf.org/thymeleaf-extras-springsecurity5">
<head>
<meta charset="utf-8">
<title>Hello~</title>
<link rel="stylesheet" href="/static/css/style.css">
</head>
<body>
   <div id="container">
       <h2>게시글 목록</h2>
       <span sec:authorize="isAuthenticated()">
           <b><font color="red">
               <span sec:authentication="principal.member.name"></span>
           </font></b>님 환영합니다.....
           <a th:href="@{/system/logout}"}>로그아웃</a>&nbsp;&nbsp;&nbsp;
           <a th:href="@{/admin/adminPage}"}>게시판 관리</a>
       </span>
```

■ 글 목록 화면 작성 – getBoardList.html

```
<thead>
 >
   번호제목작성자등록일조회수
 </thead>
 <a th:href="@{/board/getBoard(seq=${board.seq})}"
    th:text="${board.title}"></a>
   새글 등록</a>
```



■ 글 상세 화면 작성 – getBoard.html

```
<div id="container">
  <h2>글 상세 보기</h2>
  <form th:action="@{/board/updateBoard}" method="post">
     <!-- 글 수정시에 반드시 기본키를 컨트롤러에 보내야 함 -->
     <input type="hidden" name="seq" th:value="${board.seq}">
     >
          <input type="text" name="title" th:value="${board.title}">
       >
          >
             <textarea name="content" rows="10" cols="50"</pre>
                   th:text="${board.content}"></textarea>
```

■ 글 상세 화면 작성 – getBoard.html

```
>
      >
      <button type="submit">글수정</button>
        <a sec:authorize="hasRole('ROLE ADMIN')"</pre>
         th:href="@{/board/deleteBoard(seq=${board.seq})}">
          <button type="button">글삭제</button>
        </a>
        <a th:href="@{/board/getBoardList}">
          <button type="button">글목록</button>
        </a>
      </form>
```

		글쓰기	
제목			
작성자	관리자		
내용			<i>A</i>
		새글 등록	

■ 글 등록 화면 작성 – insertBoard.html

```
<div id="container">
  <h2>글쓰기</h2>
  <form th:action="@{insertBoard}" method="post">
    >
        제목
        <input type="text" name="title">
      작성자
        >
        내용
        <textarea name="content" rows="10" cols="50"></textarea>
        >
        <button type="submit">새글 등록</button>
        </form>
```

■ 시큐리티 환경 설정 – pom.xml

■ 시큐리티 설정 클래스 만들기 – SecurityConfig.java

```
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter{
   @Autowired
   private SecurityUserDetailsService userDetailsService;
   @Override
   protected void configure(HttpSecurity security) throws Exception {
       //사용자 정의 UserDetailsService 사용
       security.userDetailsService(userDetailsService);
       security.authorizeRequests()
               .antMatchers("/", "/system/**").permitAll() //인증되지 않은 모든 사용자 접근
               .antMatchers("/board/**").authenticated() //로그인한 사용자만 접근
               .antMatchers("/admin/**").hasRole("ADMIN");//ADMIN 권한을 가진 사용자만 접근
```

■ 시큐리티 설정 클래스 만들기 – SecurityConfig.java

- UserDetails 클래스 만들기
  - 1) SecurityUser 클래스 만들기

```
public class SecurityUser extends User{
    private static final long serialVersionUID = 1L;
    private Member member;
    public SecurityUser(Member member) {
        super(member.getId(), "{noop}" + member.getPassword(),
               AuthorityUtils.createAuthorityList(member.getRole().toString()));
       this.member = member;
    //인증된 회원 정보를 html에서 사용
    public Member getMember() {
       return member;
```

- UserDetails 클래스 만들기
  - 2) SecurityUserDetailsService 클래스 만들기

```
@Service
public class SecurityUserDetailsService implements UserDetailsService{
   @Autowired
    private MemberRepository memberRepo;
   @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFour
       //MemberRepository를 이용하여 조회한 회원(member) 정보를 앞에서 만든
       //SecurityUser 객체의 파라미터로 전달하여 리턴함
       Optional < Member > optional = memberRepo.findById(username);
       if(!optional.isPresent()) {
           throw new UsernameNotFoundException(username + "사용자 없음");
        }else {
           Member member = optional.get();
           return new SecurityUser(member);
```

■ Security 컨트롤러 만들기

```
@Controller
public class SecurityController {
   //로그인
   @GetMapping("/system/login")
    public void login() {}
   //접근 권한 없음 페이지
   @GetMapping("/system/accessDenied")
    public void accessDenied() {}
    //로그아웃
   @GetMapping("/system/logout")
    public void logout() {}
    //관리자 페이지
   @GetMapping("/admin/adminPage")
    public void admin() {}
```



■ 로그인 화면 작성 – login.html

```
<div id="container">
  <h2>로그인</h2>
  <form method="post">
    >
         아이디
         <input type="text" name="username">
       >비밀번호
         <input type="password" name="password">
       >
         <button type="submit">로그인</button>
         </form>
</div>
```

■ 관리자 전용 화면 작성 – adminPage.html



■ 비밀번호 암호화 하기

```
@EnableWebSecurity

public class SecurityConfig extends WebSecurityConfigurerAdapter{

//비밀번호 암호화 : PasswordEncoder 객체를 리턴하는 passwordEncoder()

@Bean

public PasswordEncoder passwordEncoder() {

return PasswordEncoderFactories.createDelegatingPasswordEncoder();

}

}
```

#### ■ 회원 등록 테스트

```
@SpringBootTest
public class BoardRepositoryTest {
    @Autowired
    private MemberRepository memberRepo;
    @Autowired
    private BoardRepository boardRepo;
    @Autowired
    private PasswordEncoder encoder;
    @Test
    public void testInsert2() {
        Member member1 = new Member();
        member1.setId("member");
        member1.setPassword(encoder.encode("member123"));
        member1.setName("김기용");
        member1.setRole(Role.ROLE_MEMBER);
        member1.setEnabled(true);
        memberRepo.save(member1);
```

```
Member member2 = new Member();
member2.setId("admin");
member2.setPassword(encoder.encode("admin123"));
member2.setName("관리자");
member2.setRole(Role.ROLE_ADMIN);
member2.setEnabled(true);
memberRepo.save(member2);
for(int i = 1; i <= 13; i++) {
    Board board = new Board();
    board.setMember(member1);
    board.setTitle(member1.getName() + "이 등록한 게시글" + i);
    board.setContent(member1.getName() + "가 등록한 게시글" + i);
    boardRepo.save(board);
for(int i = 1; i <= 3; i++) {
    Board board = new Board();
    board.setMember(member2);
    board.setTitle(member2.getName() + "이 등록한 게시글" + i);
    board.setContent(member2.getName() + "가 등록한 게시글" + i);
    boardRepo.save(board);
```

#### ■ PasswordEncoder 테스트

member_id 🦞	enabled	name	password	role
admin	1	관리자	{bcrypt}\$2a\$10\$zed0UfgCGJVZdV87XFz2ue4Ufw5z2e9emjjb	ROLE_ADMIN
member	1	김기용	{bcrypt}\$2a\$10\$gvohBzjyYYgVRLi3Sa4ZLeoEsAPBjAImFy1IAC	ROLE_MEMBER

#### {noop} 제거