

도서 쇼핑몰 웹 사이트 구축 실습 가이드

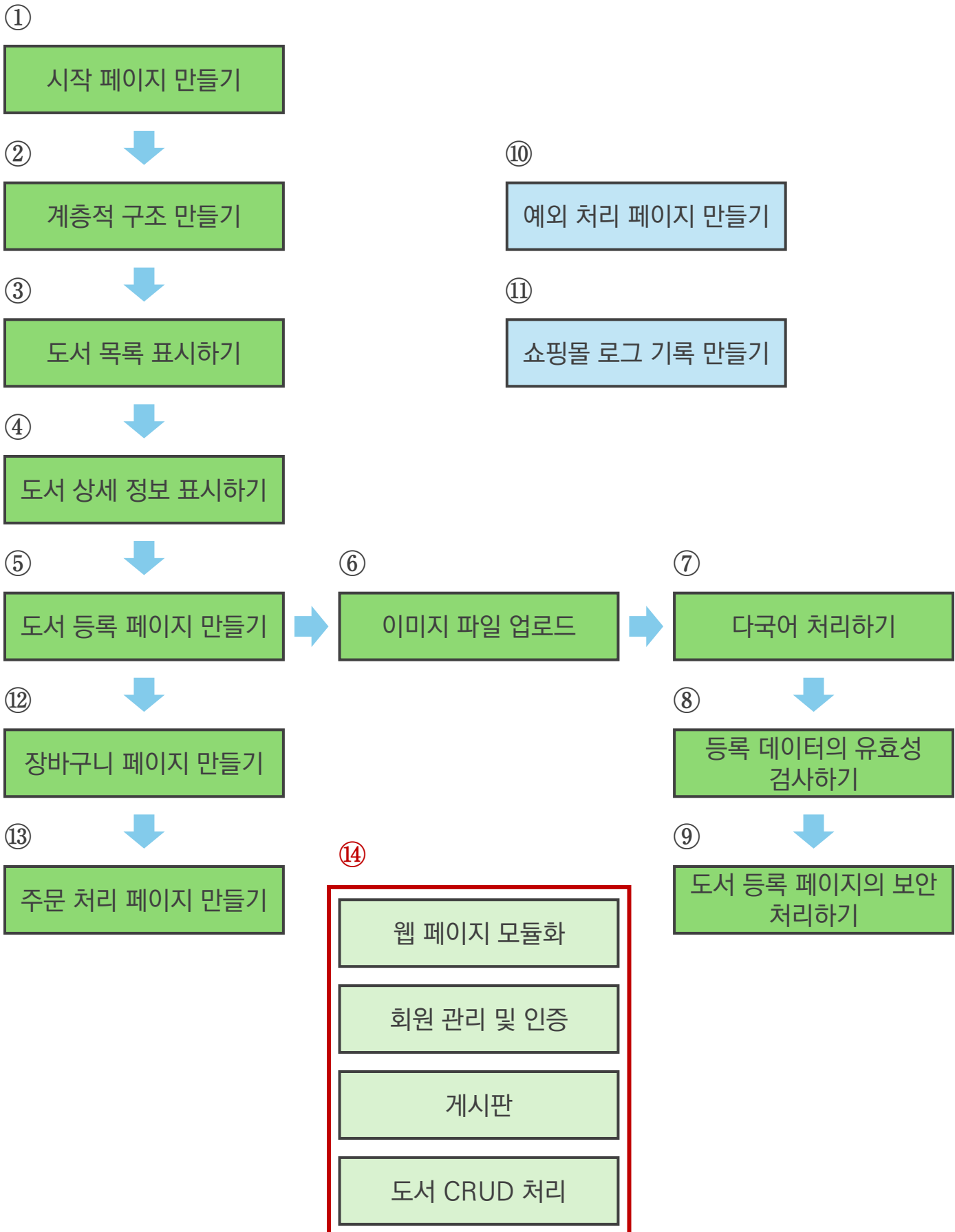
지금까지 스프링 부트의 기본 개념을 알아보고 도서 쇼핑몰 프로젝트에 직접 적용해 보았습니다.

지금까지 학습한 내용으로 충분히 웹 사이트를 구축할 수 있지만 회원 관리나 게시판 구현처럼 완성도 높은 웹 사이트를 구축하려면 pdf 파일로 제공하는 내용을 학습하시길 바랍니다.

도서 쇼핑몰 웹 사이트 구축 실습 가이드에서는 웹 사이트에 부가적으로 필요한 메뉴 구성을 포함하는 웹 페이지 모듈화, 데이터베이스를 연동하는 JPA(Java Persistence API, ORM (Object Relational Mapping) 객체 관계 매핑 기술의 표준)를 활용해 회원 가입과 수정, 탈퇴를 처리하고 게시판을 구현하며 JDBC(Java Database Connectivity)를 활용한 관리자 모드의 도서 CRUD(Create (생성), Read (읽기), Update (갱신), Delete (삭제))를 구현합니다.

다음에 소개할 가이드를 참고하여 실습을 진행하기 바랍니다.

도서 쇼핑몰 로드맵



1.1 웹 페이지 모듈화

도서 쇼핑몰의 웹 페이지 영역을 구분하여 모듈화를 구현합니다.

[실습1] 머리글 페이지와 바닥글 페이지 만들기

- 15장(pdf 파일) 2쪽을 참고하세요

1.2 회원 관리 및 인증

스프링 데이터 JPA로 데이터베이스를 연동하여 회원 가입/수정/삭제 처리를 구현합니다. 먼저 회원 관리를 위한 데이터베이스를 생성하고 엔티티 기본 구조를 만듭니다. 그리고 회원 관리 CRUD를 구현합니다.

[실습1] 회원 관리의 메뉴 및 인증 구현하기

- 15장(pdf 파일) 7쪽을 참고하세요

[실습2] 회원 관리를 위한 CRUD 구현하기

- 15장(pdf 파일) 9쪽을 참고하세요

[실습3] 회원 로그인/로그아웃 처리하기

- 15장(pdf 파일) 21쪽을 참고하세요

[실습4] 회원 관리 엔티티에 관리자 정보 등록하기

- 15장(pdf 파일) 24쪽을 참고하세요

1.3 게시판

스프링 데이터 JPA로 데이터베이스를 연동하여 게시판을 구현합니다. 게시판을 위한 데이터베이스를 생성하고 엔티티 기본 구조를 만듭니다. 그리고 CRUD를 구현합니다.

[실습1] 게시판 관리 메뉴 작성하기

- 15장(pdf 파일) 28쪽을 참고하세요

[실습2] 게시판 관리를 위한 CRUD 구현하기

- 15장(pdf 파일) 28쪽을 참고하세요

1.4 도서 CRUD 처리

스프링 데이터 JDBC로 데이터베이스를 연동하여 도서 쇼핑몰의 도서 목록 CRUD를 구현합니다.

[실습1] 데이터베이스와 테이블 생성 및 데이터 등록하기

- 15장(pdf 파일) 44쪽을 참고하세요

[실습2] JDBC 연동을 위한 환경 설정하기

- 15장(pdf 파일) 47쪽을 참고하세요

[실습3] 도서 목록 조회하기

- 15장(pdf 파일) 48쪽을 참고하세요

[실습4] 신규 도서 삽입하기

- 15장(pdf 파일) 55쪽을 참고하세요

[실습5] 도서 정보 수정하기

- 15장(pdf 파일) 56쪽을 참고하세요

[실습6] 도서 삭제하기

- 15장(pdf 파일) 65쪽을 참고하세요

도서 쇼핑몰 웹 사이트 구축

학 습 목 표

- 도서 쇼핑몰 웹 사이트 구축을 위해 웹 페이지를 모듈화합니다.
- JPA를 활용해 회원 가입/수정/탈퇴 및 게시판을 구현하고 JDBC를 활용해 도서 CRUD를 구현하여 웹 사이트를 구축합니다.

15.1

[도서 쇼핑몰] 웹 페이지 모듈화

도서 쇼핑몰의 웹 페이지 영역을 구분하여 모듈화를 구현해 봅니다.

[미리보기] 도서 쇼핑몰 웹 페이지 모듈화



실습 1 머리글 페이지와 바닥글 페이지 만들기

1 부트스트랩 JS 파일 등록 src/main/resources/static 폴더에 자바스크립트 파일을 관리하는 js 폴더가 있습니다. 여기에 부트스트랩 js 파일 bootstrap.bundle.min.js를 다운로드하여 등록 합니다.



NOTE

bootstrap.bundle.min.js 파일은 머리글에 드롭다운 메뉴를 만들기 위해 필요한 자바스크립트 파일입니다.

2 머리글 페이지 작성 /resources/templates/ 폴더에 module 폴더를 만들고 이 폴더에 header.html 파일을 생성한 뒤 다음의 내용을 작성합니다.

BookMarket/src/main/resources/templates/module/header.html

```
<header class="py-3 mb-4 border-bottom">
  <div class="container d-flex flex-wrap justify-content-center">
    <a href="/BookMarket/home" class="d-flex align-items-center link-body-emphasis
      text-body-emphasis text-decoration-none mb-3 mb-lg-0 me-lg-auto">
      <svg xmlns="http://www.w3.org/2000/svg" width="32" height="32"
        fill="currentColor" class="bi bi-book-half me-2" viewBox="0 0 16 16">
        <path d="M8.5 2.687c.654-.689 1.782-.886 3.112-.752 1.234.124 2.503.523
          3.388.893v9.923c-.918-.35-2.107-.692-3.287-.81-1.094-.111-2.278-.039-
          3.213.492M8 1.783C7.015.936 5.587.81 4.287.94c-1.514.153-3.042.672-
          3.994 1.105A5.5 5.5 0 0 0 2.5 11a5.5 5.5 0 0 0 .707 4.55c.882-.4 2.303-.881
          3.68-1.02 1.409-.142 2.59.087 3.223.877a5.5 5.5 0 0 0 .78 0c.633-.79 1.814-
          1.019 3.222-.877 1.378.139 2.8.62 3.681 1.02A5.5 5.5 0 0 0 16 13.5v-11a5.5
          5.5 0 0 0-.293-.455c-.952-.433-2.48-.952-3.994-1.105C10.413.809 8.985.936 8
          1.783"/>
        </path>
      </svg>
    </a>
  </div>
</header>
```

```

        </svg>
        <span class="fs-4">BookMarket</span>
    </a>
    <a class="nav-link dropdown-toggle" href="#" data-bs-toggle="dropdown" aria-
    expanded="false">도서목록</a>
    <ul class="dropdown-menu">
        <li><a class="dropdown-item" href="/BookMarket/books">전체도서</a></li>
        <li><a class="dropdown-item" href="/BookMarket/books/IT전문서">IT전문서
        </a></li>
        <li><a class="dropdown-item" href="/BookMarket/books/IT교육교재">IT교육
        교재</a></li>
    </ul>
    &nbsp;|&nbsp;&nbsp;&nbsp;</a>
</div>
</header>
<script src="/BookMarket/js/bootstrap.bundle.min.js" rel="stylesheet"></script>

```

3 바닥글 페이지 작성 /resources/templates/module 폴더에 footer.html 파일을 생성한 뒤 다음 내용을 작성합니다.

BookMarket/src/main/resources/templates/module/footer.html
<pre> <footer class="pt-3 mt-4 text-body-secondary border-top"> &copy; BookMarket </footer> </pre>

4 모든 웹 페이지 수정 /resources/templates 폴더에 있는 모든 *.html 파일을 다음과 같이 수정합니다.

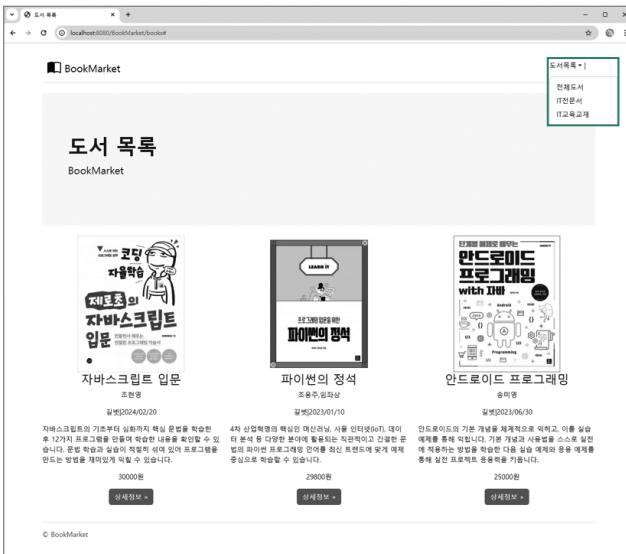
BookMarket/src/main/resources/templates/*.html
<pre> <html> ... <body> <div class="container py-4"> </pre>


```

<th:block th:replace="~/module/menu"/>/th:block>
...
<th:block th:replace="~/module/footer"/>/th:block>
</div>
</body>
</html>

```

5 프로젝트 실행 웹 브라우저에 `http://localhost:8080/BookMarket/books`를 입력하여 실행 결과를 확인합니다.

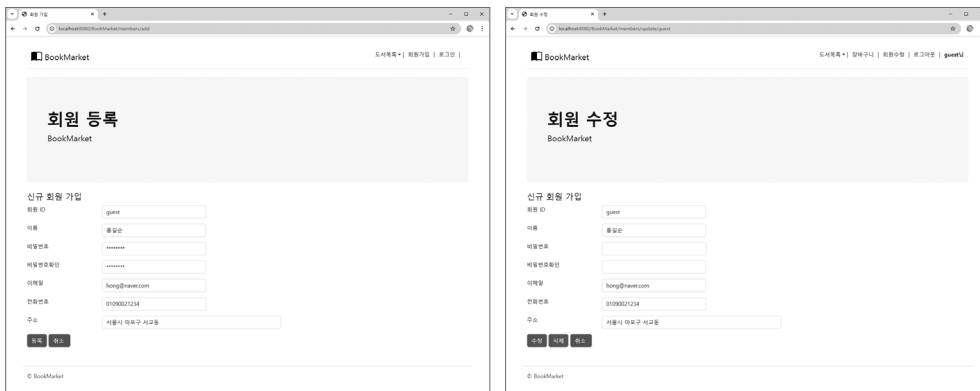


15.2

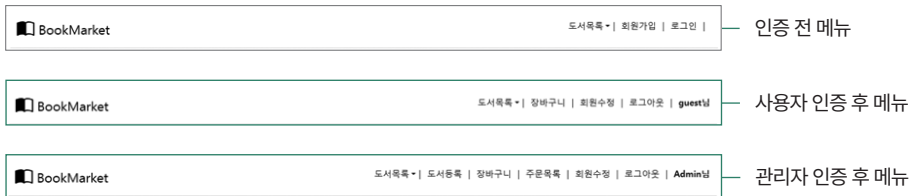
[도서쇼핑몰] 회원 관리 및 인증

스프링 데이터 JPA로 데이터베이스와 연동하여 회원 가입/수정/삭제 처리를 구현합니다. 먼저 회원 관리를 위한 데이터베이스를 생성하고 엔티티 기본 구조를 만듭니다. 그리고 회원 관리의 CRUD를 구현해 봅니다.

[미리보기] 도서 쇼핑몰의 회원 관리



[미리보기] 도서 쇼핑몰의 메뉴



1 머리글 페이지 수정 header.html 파일에 인증 전과 후의 메뉴를 변경하기 위해 다음과 같이 추가 작성합니다.

```
BookMarket/src/main/resources/templates/module/header.html

<header class="py-3 mb-4 border-bottom">

    ...

    <li><a class="dropdown-item" href="/BookMarket/books/IT교육교재">IT교육교재
    </a></li>
</ul>
<div>
    <a sec:authorize="hasRole('ROLE_ADMIN')" href="/BookMarket/books/add"
    class="nav-link">도서등록 </a>
    <a sec:authorize="isAuthenticated()" href="/BookMarket/cart" class="nav-link">
    장바구니 </a>
    <a sec:authorize="hasRole('ROLE_ADMIN')" href="/BookMarket/order/list"
    class="nav-link">주문목록 </a>
    <a sec:authorize="isAnonymous()" href="/BookMarket/members/add" class="nav-
    link">회원가입 </a>
    <a sec:authorize="isAnonymous()" href="/BookMarket/login" class="nav-link">로
    그인 </a>
    <a sec:authorize="isAuthenticated()" th:href="'/BookMarket/members/
    update/' + ${session.userLoginInfo.memberId}" class="nav-link">회원수정 </a>
    <a sec:authorize="isAuthenticated()" href="/BookMarket/logout" class="nav-
    link">로그아웃 </a>
    <b sec:authorize="isAuthenticated()"><span sec:authentication='name'></span>님
    </b>
</div>
</header>

<script src="/BookMarket/js/bootstrap.bundle.min.js" rel="stylesheet"></script>
```

2 시큐리티 수정 SecurityConfig.java 파일에 메모리상의 사용자 정보를 삭제하고 사용자 인증 후 페이지 이동을 위해 다음과 같이 작성합니다.

```

package com.springboot.config;
...
@Configuration
@EnableWebSecurity
@AllArgsConstructor
public class SecurityConfig{
    ...
    /* @Bean
    protected UserDetailsService users() {
        UserDetails admin = User.builder()
            .username("Admin")
            .password(passwordEncoder().encode("Admin1234"))
            .roles("ADMIN")
            .build();
        return new InMemoryUserDetailsManager(admin);
    } */
    @Bean
    protected SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http
        ...
        .formLogin (
            formLogin->formLogin
            .loginPage("/login") // 사용자 정의 로그인 페이지
            .loginProcessingUrl("/login")
            .defaultSuccessUrl("/books/add") // 관리자 로그인 성공 후 이동 페이지
            .defaultSuccessUrl("/order/list") // 관리자 로그인 성공 후 이동 페이지
            .defaultSuccessUrl("/") // 사용자 로그인 성공 후 이동 페이지
            .failureUrl("/loginfailed") // 로그인 실패 후 이동 페이지
            .usernameParameter("username")
            .passwordParameter("password")
        )
        ...
        return http.build();
    }
}

```

❶ /* */로 주석 처리한 부분을 삭제합니다.

실습 2 회원 관리를 위한 CRUD 구현하기

스프링 데이터 JPA를 적용해 회원 등록, 수정, 삭제 등의 처리를 하기 위해 회원 관리의 공통 모듈을 만듭니다.

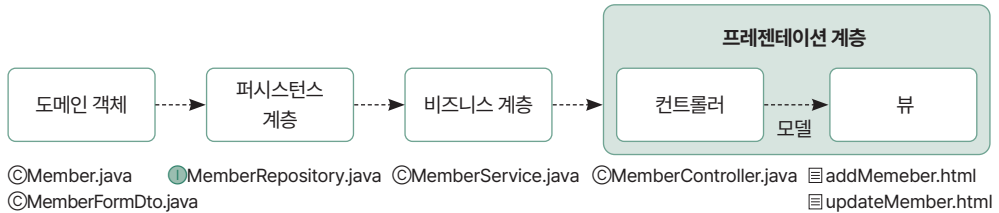


그림 15-1 도서 쇼핑몰 회원 관리의 계층적 구조

1 사용자 역할(ROLE) 설정 com.springboot.domain 패키지에 Role.java 파일을 생성하고 다음 내용을 작성합니다.

```
BookMarket/src/main/java/com/springboot/domain/Role.java

package com.springboot.domain;

public enum Role {

    USER, ADMIN

}
```

2 회원 관리 엔티티 작성 com.springboot.domain 패키지에 Member 클래스를 생성하고 다음 내용을 작성합니다.

```
BookMarket/src/main/java/com/springboot/domain/Member.java

package com.springboot.domain;

import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import org.springframework.security.crypto.password.PasswordEncoder;
import jakarta.persistence.*;

@Entity
```

```

@Table(name="member")
@Getter @Setter
@NoArgsConstructor
public class Member {
    @Id
    @Column(name="num")
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long num;
    @Column(unique = true)
    private String memberId;
    private String password;
    private String name;
    private String phone;
    private String email;
    private String address;
    @Enumerated(EnumType.STRING)
    private Role role;

    public static Member createMember(MemberFormDto memberFormDto, PasswordEncoder
        passwordEncoder) {
        Member member = new Member();
        member.setMemberId(memberFormDto.getMemberId());
        member.setName(memberFormDto.getName());
        member.setPhone(memberFormDto.getPhone());
        member.setEmail(memberFormDto.getEmail());
        member.setAddress(memberFormDto.getAddress());
        String password = passwordEncoder.encode(memberFormDto.getPassword());
        member.setPassword(password);
        member.setRole(Role.USER);
        return member;
    }
}

```

3 회원 관리 도메인 객체 작성 com.springboot.domain 패키지에 MemberFormDto 클래스를 생성하고 다음 내용을 작성합니다.

BookMarket/src/main/java/com/springboot/domain/MemberFormDto.java

```

package com.springboot.domain;

import lombok.Getter;
import lombok.Setter;
import org.hibernate.validator.constraints.Length;
import jakarta.validation.constraints.Email;
import jakarta.validation.constraints.NotBlank;
import jakarta.validation.constraints.NotEmpty;

@Getter @Setter
public class MemberFormDto {
    @NotBlank(message = "아이디는 필수 입력 값입니다.")
    private String memberId;
    @NotEmpty(message = "비밀번호는 필수 입력 값입니다.")
    @Length(min=8, max=16, message = "비밀번호는 6자 이상, 16자 이하로 입력해주세요")
    private String password;
    @NotBlank(message = "이름은 필수 입력 값입니다.")
    private String name;
    @NotBlank(message = "연락처는 필수 입력 값입니다.")
    private String phone;
    @NotEmpty(message = "이메일은 필수 입력 값입니다.")
    @Email(message = "이메일 형식으로 입력해주세요.")
    private String email;
    @NotEmpty(message = "주소는 필수 입력 값입니다.")
    private String address;
}

```

4 회원 관리 저장소 객체 작성 com.springboot.repository 패키지에 MemberRepository 인터페이스를 생성하고 다음 내용을 작성합니다.

BookMarket/src/main/java/com/springboot/repository/MemberRepository.java

```

package com.springboot.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import com.springboot.domain.Member;

```

```

@Repository
public interface MemberRepository extends JpaRepository<Member, Long> {
    Member findByMemberId(String memberId);
}

```

5 회원 관리 서비스 객체 작성 com.springboot.service 패키지에 UserDetailsService 인터페이스를 구현한 MemberService 클래스를 생성하고 다음 내용을 작성합니다.

BookMarket/src/main/java/com/springboot/service/MemberService.java

```

package com.springboot.service;
import lombok.RequiredArgsConstructor;
import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.stereotype.Service;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import com.springboot.domain.Member;
import com.springboot.repository.MemberRepository;
import jakarta.transaction.Transactional;

@Service
@Transactional
@RequiredArgsConstructor
public class MemberService implements UserDetailsService {
    private final MemberRepository memberRepository;

    public Member saveMember(Member member) {           // 회원 정보 저장하기
        validateDuplicateMember(member);
        return memberRepository.save(member);
    }

    public Member getMemberById(String memberId) {       // 회원 정보 가져오기
        Member member = memberRepository.findByMemberId(memberId);
        return member;
    }

    public void deleteMember(String memberId) {          // 회원 삭제하기
        Member member = memberRepository.findByMemberId(memberId);
        memberRepository.deleteById(member.getNum());
    }
}

```



```

    }
    private void validateDuplicateMember(Member member) {    // 회원 id 중복 체크하기
        Member findMember = memberRepository.findByMemberId(member.getMemberId());
        if(findMember != null) {
            throw new IllegalStateException("이미 가입된 회원입니다.");
        }
    }
    // 인증 시 회원 정보 가져오기
    @Override
    public UserDetails loadUserByUsername(String id) throws
        UsernameNotFoundException {
        Member member = memberRepository.findByMemberId(id);
        if(member == null) {
            throw new UsernameNotFoundException(id);
        }
        return User.builder()
            .username(member.getMemberId())
            .password(member.getPassword())
            .roles(member.getRole().toString())
            .build();
    }
}

```

6 회원 관리 컨트롤러 작성 com.springboot.controller 패키지에 MemberController 클래스를 생성하고 다음의 내용을 작성합니다.

BookMarket/src/main/java/com/springboot/controller/MemberController.java

```

package com.springboot.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;

```

```

import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import com.springboot.domain.Member;
import com.springboot.domain.MemberFormDto;
import com.springboot.service.MemberService;
import jakarta.validation.Valid;
@Controller
@RequestMapping(value = "/members")
public class MemberController {
    @Autowired
    MemberService memberService;
    @Autowired
    PasswordEncoder passwordEncoder;
    // 신규 회원 등록 페이지 출력하기
    @GetMapping(value = "/add")
    public String requestAddMemberForm(Model model) {
        model.addAttribute("memberFormDto", new MemberFormDto());
        return "member/addMember";
    }
    // 신규 회원 등록하기
    @PostMapping(value = "/add")
    public String submitAddNewMember(@Valid MemberFormDto memberFormDto,
        BindingResult bindingResult, Model model) {
        if(bindingResult.hasErrors()) {
            return "member/addMember";
        }
        try {
            Member member = Member.createMember(memberFormDto, passwordEncoder);
            memberService.saveMember(member);
        } catch (IllegalStateException e) {
            model.addAttribute("errorMessage", e.getMessage());
            return "member/addMember";
        }
        return "redirect:/members";
    }
    // 회원 정보 수정 페이지 출력하기

```

```

@GetMapping(value = "/update/{memberId}")
public String requestUpdateMemberForm(@PathVariable(name = "memberId") String
    memberId, Model model) {
    Member member = memberService.getMemberById(memberId);
    model.addAttribute("memberFormDto", member);
    return "member/updateMember";
}

// 회원 정보 수정하기
@PostMapping(value = "/update")
public String submitUpdateMember(@Valid MemberFormDto memberFormDto,
    BindingResult bindingResult, Model model) {
    if(bindingResult.hasErrors()) {
        return "member/updateMember";
    }
    try {
        Member member = Member.createMember(memberFormDto, passwordEncoder);
        memberService.saveMember(member);
    } catch (IllegalStateException e) {
        model.addAttribute("errorMessage", e.getMessage());
        return "member/addMember";
    }
    return "redirect:/members";
}

// 회원 정보 삭제하기
@GetMapping("/delete/{memberId}")
public String deleteMember(@PathVariable(name = "memberId") String memberId) {
    memberService.deleteMember(memberId);
    return "redirect:/logout";
}

// 회원 가입 및 인증 시 인사말 페이지로 이동하기
@GetMapping
public String requestMain() {
    return "redirect:/";
}
}

```

7 회원 가입 페이지 작성 /resources/templates/ 폴더에 member 폴더를 만듭니다. 이 폴더에 addMemeber.html 파일을 생성하여 다음의 내용을 작성합니다.

BookMarket/src/main/resources/templates/member/addMemeber.html

```
<html>
<head>
  <title>회원 가입</title>
  <link href="/BookMarket/css/bootstrap.min.css" rel="stylesheet">
</head>
<script type="text/javascript">
  function checkPasswd() {
    // 비밀번호와 비밀번호 확인 항목이 일치하는지 검사
    if(document.member.password.value != document.member.password_confirm.value) {
      alert("비밀번호를 확인해 주세요");
      return;
    }
    document.member.submit();
  }
</script>
<body>
<div class="container py-4">
  <th:block th:replace="~/module/header"></th:block>
  <div class="p-5 mb-4 bg-body-tertiary rounded-3">
    <div class="container-fluid py-5">
      <h1 class="display-5 fw-bold">회원 등록</h1>
      <p class="col-md-8 fs-4">BookMarket</p>
    </div>
  </div>
  <div class="row align-items-md-stretch">
    <div class="alert alert-danger th:if="${errorMessage!=null}">
      [[${errorMessage}]]<br/>
    </div>
    <form th:object="${memberFormDto}" name="member" action="/BookMarket/members/add" method="post">
      <legend>신규 회원 가입</legend>
      <div class="mb-3 row">
```

```

<label class="col-sm-2 control-label">회원 ID</label>
<div class="col-sm-3">
  <input type="text" name="memberId" class="form-control"
    th:field="*{memberId}"/>
</div>
</div>
<div class="mb-3 row">
  <label class="col-sm-2 control-label">이름</label>
  <div class="col-sm-3">
    <input type="text" name="name" class="form-control" th:field="*{name}"/>
  </div>
  <div class="col-sm-6">
    <p class="text-danger" th:errors="*{name}"/>
  </div>
</div>
<div class="mb-3 row">
  <label class="col-sm-2 control-label">비밀번호</label>
  <div class="col-sm-3">
    <input type="text" name="password" class="form-control"
      th:field="*{password}"/>
  </div>
  <div class="col-sm-6">
    <p class="text-danger" th:errors="*{password}"/>
  </div>
</div>
<div class="mb-3 row">
  <label class="col-sm-2 control-label">비밀번호 확인</label>
  <div class="col-sm-3">
    <input type="text" name="password_confirm" class="form-control"/>
  </div>
</div>
<div class="mb-3 row">
  <label class="col-sm-2 control-label">이메일</label>
  <div class="col-sm-3">
    <input type="text" name="email" class="form-control"
      th:field="*{email}"/>

```

```

    </div>
    <div class="col-sm-6">
        <p class="text-danger" th:errors="*{email}">/>
    </div>
</div>
<div class="mb-3 row">
    <label class="col-sm-2 control-label">전화번호</label>
    <div class="col-sm-3">
        <input type="text" name="phone" class="form-control"
            th:field="*{phone}">/>
    </div>
    <div class="col-sm-6">
        <p class="text-danger" th:errors="*{phone}">/>
    </div>
</div>
<div class="mb-3 row">
    <label class="col-sm-2 control-label">주소</label>
    <div class="col-sm-5">
        <input type="text" name="address" cols="50" rows="2" class="form-
            control" th:field="*{address}">/>
    </div>
</div>
<div class="mb-3 row">
    <div class="col-sm-offset-2 col-sm-10">
        <a th:href="javascript:checkPasswd()" class="btn btn-primary">등록</a>
        <input type="reset" class="btn btn-secondary" value ="취소"/>
    </div>
</div>
</form>
<th:block th:replace="~{/module/footer}"×/th:block>
</div>
</body>
</html>

```

8 회원 수정 페이지 작성 /resources/templates/member 폴더에 updateMember.html 파일을 생성하여 다음의 내용을 작성합니다.

BookMarket/src/main/resources/templates/member/updateMember.html

```
<html>
<head>
  <title>회원 수정</title>
  <link href="/BookMarket/css/bootstrap.min.css" rel="stylesheet">
</head>
<script type="text/javascript">
  function checkPasswd() {
    // 비밀번호와 비밀번호 확인 항목이 일치하는지 검사
    if(document.member.password.value != document.member.password_confirm.value) {
      alert("비밀번호를 확인해 주세요");
      return;
    }
    document.member.submit();
  }
</script>
<body>
<div class="container py-4">
  <th:block th:replace="~/module/header">X</th:block>
  <div class="p-5 mb-4 bg-body-tertiary rounded-3">
    <div class="container-fluid py-5">
      <h1 class="display-5 fw-bold">회원 수정</h1>
      <p class="col-md-8 fs-4">BookMarket</p>
    </div>
  </div>
  <div class="row align-items-md-stretch">
    <form th:object="${memberFormDto}" name="member" action="/BookMarket/members/
    update" method="post">
      <legend>신규 회원 가입</legend>
      <div class="mb-3 row">
        <label class="col-sm-2 control-label">회원 ID</label>
        <div class="col-sm-3">
          <input type="text" name="memberId" class="form-control"
            th:field="*{memberId}" readonly/>

```

```

    </div>
</div>
<div class="mb-3 row">
    <label class="col-sm-2 control-label">이름</label>
    <div class="col-sm-3">
        <input type="text" name="name" class="form-control"
            th:field="*{name}"/>
    </div>
    <div class="col-sm-6">
        <p class="text-danger" th:errors="*{name}"/>
    </div>
</div>
<div class="mb-3 row">
    <label class="col-sm-2 control-label">비밀번호</label>
    <div class="col-sm-3">
        <input type="text" name="password" class="form-control"/>
    </div>
    <div class="col-sm-6">
        <p class="text-danger" th:errors="*{password}"/>
    </div>
</div>
<div class="mb-3 row">
    <label class="col-sm-2 control-label">비밀번호 확인</label>
    <div class="col-sm-3">
        <input type="text" name="password_confirm" class="form-control"/>
    </div>
</div>
<div class="mb-3 row">
    <label class="col-sm-2 control-label">이메일</label>
    <div class="col-sm-3">
        <input type="text" name="email" class="form-control"
            th:field="*{email}"/>
    </div>
    <div class="col-sm-6">
        <p class="text-danger" th:errors="*{email}"/>
    </div>
</div>

```



```

<div class="mb-3 row">
  <label class="col-sm-2 control-label">전화번호</label>
  <div class="col-sm-3">
    <input type="text" name="phone" class="form-control"
      th:field="*{phone}"/>
  </div>
  <div class="col-sm-6">
    <p class="text-danger" th:errors="*{phone}"/>
  </div>
</div>
<div class="mb-3 row">
  <label class="col-sm-2 control-label">주소</label>
  <div class="col-sm-5">
    <input type="text" name="address" cols="50" rows="2" class="form-
      control" th:field="*{address}"/>
  </div>
</div>
<div class="mb-3 row">
  <div class="col-sm-offset-2 col-sm-10">
    <a th:href="'javascript:checkPasswd()'" class="btn btn-success">수정</a>
    <a th:href="@{'/members/delete/' + ${memberId}]" class="btn btn-danger">
      삭제</a>
    <input type="reset" class="btn btn-Secondary" value ="취소"/>
  </div>
</div>
</form>
<th:block th:replace="~/module/footer"></th:block>
</div>
</body>
</html>

```

실습 3 회원 로그인/로그아웃 처리하기

1 로그인 후 인사말 컨트롤러 작성 com.springboot.controller 패키지에 HomeController 클래스를 생성하고 다음의 내용을 작성합니다.

```

package com.springboot.controller;
import org.springframework.stereotype.Controller;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.userdetails.User;
import org.springframework.ui.Model;
import com.springboot.domain.Member;
import com.springboot.service.MemberService;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpSession;
@Controller
public class HomeController {
    @Autowired
    private MemberService memberService;
    // 웹 요청 URL이 /인 경우에 호출하는 메서드
    @RequestMapping("/")
    public String welcome(Model model, Authentication authentication, HttpServletRequest
        httpRequest) {
        if (authentication == null) // 인증 전 처리
            return "welcome";
        // 인증 후 처리
        User user = (User) authentication.getPrincipal();
        String userId = user.getUsername();
        if(userId == null)
            return "redirect:/login";
        Member member = memberService.getMemberById(userId);
        HttpSession session = httpRequest.getSession(true);
        session.setAttribute("userLoginInfo", member); // 사용자 정보의 세션 등록
        return "welcome";
    }
}

```

2 인사말 페이지 작성 welcome.html 파일을 생성하고 다음의 내용을 작성합니다.

BookMarket/src/main/resources/templates/welcome.html

```
<html>
<head>
  <meta charset='UTF-8'>
  <title>Welcome</title>
  <link href='https://getbootstrap.com/docs/5.3/dist/css/bootstrap.min.css'
    rel='stylesheet'></head>
<body>
  <div class='container py-4'>
    <th:block th:replace="~/module/header"></th:block>
    <div class='p-5 mb-4 bg-body-tertiary rounded-3'>
      <div class='container-fluid py-5'>
        <h1 class='display-5 fw-bold'>도서 쇼핑물에 오신 것을 환영합니다</h1>
        <p class='col-md-8 fs-4'>BookMarket</p>
      </div>
    </div>
    <div class='row align-items-md-stretch text-center'>
      <div class='col-md-12'>
        <div class='h-100 p-5'>
          <h2>Welcome to Web Market!</h2>
        </div>
      </div>
    </div>
    <th:block th:replace="~/module/footer"></th:block>
  </div>
</body>
</html>
```

3 로그인 요청 메서드 수정 com.springboot.controller 패키지의 LoginController 클래스에 다음의 내용을 추가 작성합니다.

BookMarket/src/main/java/com/springboot/controller/LoginController.java

```
package com.springboot.controller;
import org.springframework.stereotype.Controller;
```

```

import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpSession;
@Controller
public class LoginController {
    ...
    // 로그아웃 처리
    @GetMapping("/logout")
    public String logout(Model model, HttpServletRequest request) {
        HttpSession session = request.getSession(false); // 세션 존재 여부 확인
        if(session != null) {
            session.invalidate(); // 세션 삭제
        }
        return "login";
    }
}

```

실습 4 회원 관리 엔티티에 관리자 정보 등록하기

1 관리자 정보 등록 com.springboot 패키지의 BookMarketApplication 클래스에 다음의 내용을 추가 작성합니다.

BookMarket/src/main/java/com/springboot/BookMarketApplication.java

```

package com.springboot;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import com.springboot.domain.Member;
import com.springboot.domain.Role;
import com.springboot.service.MemberService;

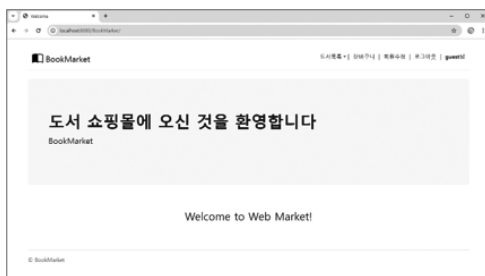
```

```

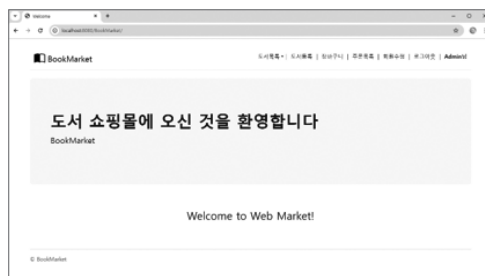
@SpringBootApplication
public class BookMarketApplication {
    public static void main(String[] args) {
        SpringApplication.run(BookMarketApplication.class, args);
    }
    // 관리자 정보를 Member 엔티티에 등록
    @Bean
    public CommandLineRunner run(MemberService memberService) throws Exception {
        return (String[] args) -> {
            Member member = new Member();
            member.setMemberId("Admin");
            member.setName("관리자");
            member.setPhone("");
            member.setEmail("");
            member.setAddress("");
            String password = new BCryptPasswordEncoder().encode("Admin1234");
            member.setPassword(password);
            member.setRole(Role.ADMIN);
            memberService.saveMember(member);
        }
    }
}

```

2 프로젝트 실행 웹 브라우저에 <http://localhost:8080/BookMarket/>을 입력하여 실행합니다. 관리자로 로그인하거나 회원 가입을 한 후 로그인하여 회원 수정/삭제를 실행해 봅니다.



▲ 사용자 인증 후



▲ 관리자 인증 후

BookMarket 도서목록 > 회원가입 > 로그인 >

회원 등록

BookMarket

신규 회원 가입

회원 ID:

이름:

비밀번호:

비밀번호확인:

이메일:

전화번호:

주소:

© BookMarket

BookMarket 도서목록 > 관리자 > 회원목록 > 로그인 > guest

도서 쇼핑물에 오신 것을 환영합니다

BookMarket

신규 회원 가입

회원 ID:

이름:

비밀번호:

비밀번호확인:

이메일:

전화번호:

주소:

© BookMarket

BookMarket 도서목록 > 회원가입 > 로그인 >

회원 등록

BookMarket

신규 회원 가입

회원 ID:

이름: 이름은 중간 글자도 입력하!

비밀번호: 비밀번호는 중간 글자도 입력하!

비밀번호확인: 비밀번호를 2번 이상 100% 일치로 입력해야하!

이메일: 이메일은 필수 입력도입하!

전화번호: 입력가능한 필수 입력도입하!

주소:

© BookMarket

BookMarket 도서목록 > 회원가입 > 로그인 >

회원 등록

BookMarket

회원 가입과 비밀번호를 설정합니다.

신규 회원 가입

회원 ID:

이름:

비밀번호:

비밀번호확인:

이메일:

전화번호:

주소:

© BookMarket

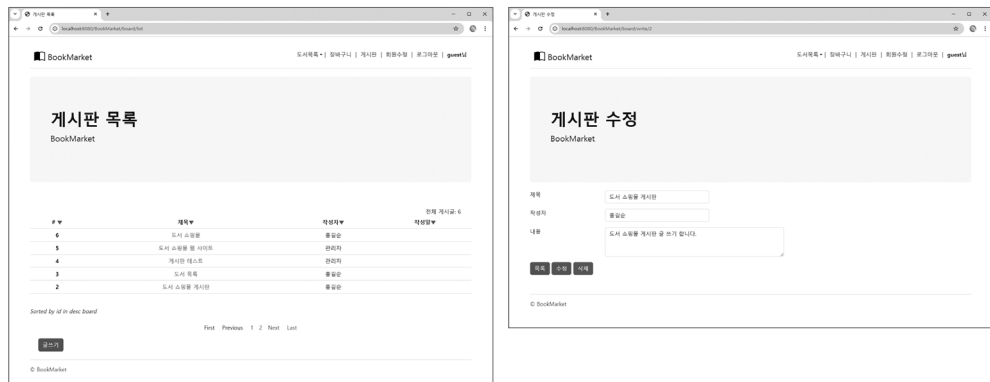
15.3

[도서쇼핑몰] 게시판

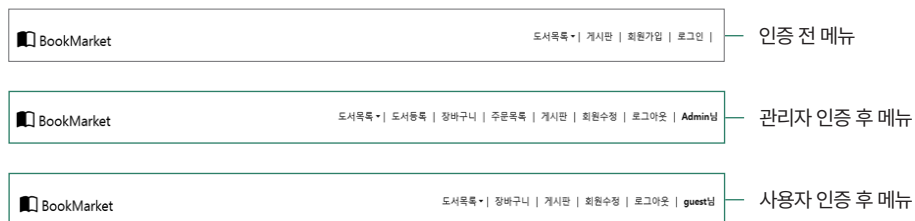
15

스프링 데이터 JPA로 데이터베이스와 연동하여 게시판을 구현합니다. 먼저 게시판을 위한 데이터베이스를 생성하고 엔티티 기본 구조를 만듭니다. 그리고 게시판의 CRUD를 구현해 봅니다.

[미리보기] 도서 쇼핑몰의 게시판



[미리보기] 도서 쇼핑몰의 메뉴



실습 1 게시판 관리 메뉴 작성하기

1 머리글 페이지 수정 인증 전과 후의 메뉴를 변경하기 위해 header.html 파일에 다음과 같이 추가 작성합니다.

BookMarket/src/main/resources/templates/module/header.html

```
<header class="py-3 mb-4 border-bottom">  
...  
  
    <a sec:authorize="hasRole('ROLE_ADMIN')" href="/BookMarket/order/list"  
        class="nav-link">주문목록 &nbsp;&nbsp;&|&nbsp;&nbsp;&/a>  
    <a href="/BookMarket/board/list" class="nav-link">게시판 &nbsp;&nbsp;&|&nbsp;&nbsp;&/a>  
    <a sec:authorize="isAnonymous()" href="/BookMarket/members/add" class="nav-  
link">회원가입 &nbsp;&nbsp;&|&nbsp;&nbsp;&/a>  
...  
  
<script src="/BookMarket/js/bootstrap.bundle.min.js" rel="stylesheet"></script>
```

실습 2 게시판 관리를 위한 CRUD 구현하기

스프링 데이터 JPA를 적용해 회원 관리 등록, 수정, 삭제 등의 처리를 하기 위해 회원 관리의 공통 모듈을 만듭니다.

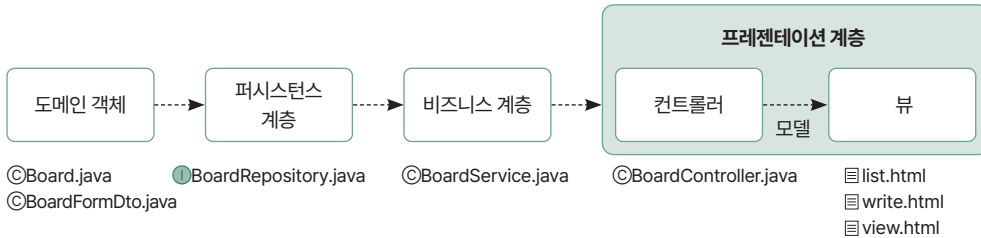


그림 15-2 도서 쇼핑몰 게시판의 계층적 구조

1 게시판 관리 엔티티 생성 com.springboot.domain 패키지에 Board 클래스를 생성하고 다음 내용을 작성합니다.


```

package com.springboot.domain;

import java.time.LocalDateTime;

import org.springframework.data.annotation.CreatedDate;
import org.springframework.data.annotation.LastModifiedDate;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.EntityListeners;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.Id;

import lombok.Builder;
import lombok.Getter;
import lombok.NoArgsConstructor;
import org.springframework.data.jpa.domain.support.AuditingEntityListener;
import lombok.AccessLevel;

@Getter
@Entity
@NoArgsConstructor(access = AccessLevel.PROTECTED)
@EntityListeners(AuditingEntityListener.class)
public class Board {

    @Id
    @GeneratedValue
    private Long id;
    @Column(length = 10, nullable = false)
    private String writerid;
    @Column(length = 10, nullable = false)
    private String writer;
    @Column(length = 100, nullable = false)
    private String title;
    @Column(columnDefinition = "TEXT", nullable = false)
    private String content;
    @CreatedDate
    @Column(updatable = false)
    private LocalDateTime createdDate;
    @LastModifiedDate
    private LocalDateTime modifiedDate;

    @Builder

```

```

    public Board(Long id, String writerid, String writer, String title, String content) {
        this.id = id;
        this.writerid = writerid;
        this.writer = writer;
        this.title = title;
        this.content = content;
    }
}

```

2 게시판 관리 도메인 객체 생성 com.springboot.domain 패키지에 BoardFormDto 클래스를 생성하고 다음 내용을 작성합니다.

BookMarket/src/main/java/com/springboot/domain/BoardFormDto.java

```

package com.springboot.domain;
import lombok.*;
import java.time.LocalDateTime;

@Getter
@Setter
@ToString
@NoArgsConstructor
public class BoardFormDto {
    private Long id;
    private String writerid;
    private String writer;
    private String title;
    private String content;
    private LocalDateTime createdDate;
    private LocalDateTime modifiedDate;
    public Board toEntity() {
        Board build = Board.builder()
            .id(id)
            .writerid(writerid)
            .writer(writer)
            .title(title)
            .content(content)

```

```

        .build();
        return build;
    }

    @Builder
    public BoardFormDto(Long id, String writerid, String writer, String title, String
        content, LocalDateTime createdAt, LocalDateTime modifiedAt) {
        this.id = id;
        this.writerid = writerid;
        this.writer = writer;
        this.title = title;
        this.content = content;
        this.createdAt = createdAt;
        this.modifiedAt = modifiedAt;
    }
}

```

3 게시판 관리 저장소 객체 생성 com.springboot.repository 패키지에 BoardRepository 인터페이스를 생성하고 다음 내용을 작성합니다.

BookMarket/src/main/java/com/springboot/repository/BoardRepository.java

```

package com.springboot.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import com.springboot.domain.Board;
public interface BoardRepository extends JpaRepository<Board, Long> {

}

```

4 게시판 관리 서비스 객체 생성 com.springboot.service 패키지에 BoardService 클래스를 생성하고 다음 내용을 작성합니다.

BookMarket/src/main/java/com/springboot/service/BoardService.java

```

package com.springboot.service;
import java.util.List;

```

```

import java.util.ArrayList;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.stereotype.Service;
import com.springboot.domain.Board;
import com.springboot.domain.BoardFormDto;
import com.springboot.repository.BoardRepository;
import jakarta.transaction.Transactional;

@Service
public class BoardService {

    @Autowired
    private BoardRepository boardRepository;
    // 게시글 등록/수정하기
    @Transactional
    public Long savePost(BoardFormDto boardDto) {
        return boardRepository.save(boardDto.toEntity()).getId();
    }
    // 전체 게시글 가져오기
    @Transactional
    public List<BoardFormDto> getBoardList() {
        List<Board> boardList = boardRepository.findAll();
        List<BoardFormDto> boardDtoList = new ArrayList<>();
        for(Board board : boardList) {
            BoardFormDto boardDto = BoardFormDto.builder()
                .id(board.getId())
                .writerid(board.getWriterid())
                .writer(board.getWriter())
                .title(board.getTitle())
                .content(board.getContent())
                .createdDate(board.getCreatedDate())
                .build();
            boardDtoList.add(boardDto);
        }
        return boardDtoList;
    }
}

```

```

    }
    // 게시글 가져오기
    public Board getBoardById(Long id) {
        Board board = boardRepository.findById(id).get();
        return board;
    }
    // 게시글 삭제하기
    public void deleteBoardById(Long id) {
        boardRepository.deleteById(id);
    }
    // 전체 게시글의 목록 개수, 정렬 가져오기
    public Page<Board> listAll(int pageNum, String sortField, String sortDir) {
        int pageSize = 5;
        Pageable pageable = PageRequest.of(pageNum - 1, pageSize, sortDir.
            equals("asc") ? Sort.by(sortField).ascending() : Sort.by(sortField).
            descending());
        return boardRepository.findAll(pageable);
    }
}

```

5 게시판 컨트롤러 작성 com.springboot.controller 패키지에 BoardController 클래스를 생성하고 다음의 내용을 작성합니다.

BookMarket/src/main/java/com/springboot/controller/BoardController.java

```

package com.springboot.controller;

import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;

```

```

import org.springframework.web.bind.annotation.RequestParam;
import com.springboot.domain.Board;
import com.springboot.domain.BoardFormDto;
import com.springboot.domain.Member;
import com.springboot.service.BoardService;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpSession;
import jakarta.validation.Valid;
@Controller
@RequestMapping(value = "/board")
public class BoardController {
    @Autowired
    private BoardService boardService;
    // 전체 게시글 목록 가져오기
    @GetMapping("/list")
    public String viewHomePage(Model model) {
        return viewPage( 1, "id", "desc",model );
    }
    // 전체 게시글 가져오기
    @GetMapping("/page")
    public String viewPage(@RequestParam("pageNum") int pageNum,
        @RequestParam("sortField") String sortField, @RequestParam("sortDir")
        String sortDir, Model model) {
        Page<Board> page = boardService.listAll(pageNum, sortField, sortDir);
        List<Board> listBoard = page.getContent();
        model.addAttribute("currentPage", pageNum);
        model.addAttribute("totalPages", page.getTotalPages());
        model.addAttribute("totalItems", page.getTotalElements());
        model.addAttribute("sortField", sortField);
        model.addAttribute("sortDir", sortDir);
        model.addAttribute("reverseSortDir", sortDir.equals("asc") ? "desc" : "asc");
        model.addAttribute("boardList", listBoard);
        return "board/list";
    }
    // 게시글 글쓰기 페이지 출력하기
    @GetMapping("/write")
    public String post() {

```

```

        return "board/write";
    }

    // 게시물 글쓰기 저장하기
    @PostMapping("/write")
    public String write(BoardFormDto boardDto) {
        boardService.savePost(boardDto);
        return "redirect:/board/list";
    }

    // 게시물 상세 보기
    @GetMapping("/view/{id}")
    public String requestUpdateMemberForm(@PathVariable(name = "id") Long id,
        HttpServletRequest httpRequest, Model model){
        Board board = boardService.getBoardById(id);
        model.addAttribute("boardFormDto", board);
        HttpSession session = httpRequest.getSession(true);
        Member member = (Member) session.getAttribute("userLoginInfo");
        model.addAttribute("buttonOk", false);
        if(member != null && board.getWriterid().equals(member.getMemberId())) {
            model.addAttribute("buttonOk", true);
        }
        return "board/view";
    }

    // 게시물 수정하기
    @PostMapping("/update")
    public String submitUpdateMember(@Valid BoardFormDto boardDto, BindingResult
        bindingResult, Model model) {
        if(bindingResult.hasErrors())
            return "board/view";
        try {
            boardService.savePost(boardDto);
        } catch (IllegalStateException e) {
            model.addAttribute("errorMessage", e.getMessage());
            return "board/view";
        }
        return "redirect:/board/list";
    }
}

```

```
// 게시물 삭제하기
@GetMapping("/delete/{id}")
public String deleteOrder(@PathVariable(name = "id") Long id) {
    boardService.deleteBoardById(id);
    return "redirect:/board/list";
}
}
```

6 게시판 목록 페이지 작성 /resources/templates/ 폴더에 board 폴더를 만듭니다. 이 폴더에 list.html 파일을 생성하여 다음의 내용을 작성합니다.

BookMarket/src/main/resources/templates/board/list.html

```
<html>
<head>
    <meta charset="UTF-8"/>
    <title>게시판 목록</title>
    <link href="/BookMarket/css/bootstrap.min.css" rel="stylesheet">
</head>
<style>
    a {
        text-decoration:none;
    }
</style>
<body>
    <div class="container py-4">
        <th:block th:replace="~/module/header"></th:block>
        <div class="p-5 mb-4 bg-body-tertiary rounded-3">
            <div class="container-fluid py-5">
                <h1 class="display-5 fw-bold">게시판 목록</h1>
                <p class="col-md-8 fs-4">BookMarket</p>
            </div>
        </div>
        <div class="row align-items-md-stretch">
            <div style="padding-top: 50px">
```



```

<div class="text-end" style="padding-right:30px">전체 게시물:
[[${totalItems}]]</div>
<table class="table">
<thead class="thead-light">
<tr class="text-center">                                // 정렬 표시
  <th>#<a th:href="@{'/board/page?pageNum=' + ${currentPage}
+ '&sortField=id&sortDir=' + ${reverseSortDir}}">X<span
th:text="${reverseSortDir} == 'asc'? '▼': '▲'">X</a>X</th>
  <th>제목<a th:href="@{'/board/page?pageNum=' + ${currentPage}
+ '&sortField=title&sortDir=' + ${reverseSortDir}}">X<span
th:text="${reverseSortDir} == 'asc'? '▼': '▲'">X</a>X</th>
  <th>작성자<a th:href="@{'/board/page?pageNum=' + ${currentPage}
+ '&sortField=writer&sortDir=' + ${reverseSortDir}}">X<span
th:text="${reverseSortDir} == 'asc'? '▼': '▲'">X</a>X</th>
  <th>작성일<a th:href="@{'/board/page?pageNum=' + ${currentPage}
+ '&sortField=createdDate&sortDir=' + ${reverseSortDir}}">X<span
th:text="${reverseSortDir} == 'asc'? '▼': '▲'">X</a>X</th>
</tr>
</thead>
<tbody>
<tr class="text-center" th:each="board : ${boardList}">
<th scope="row">
  <span th:text="${board.id}">X</span>
</th>
<th>
  <a th:href="@{'/board/view/' + ${board.id}}">
  <span th:text="${board.title}">X</span>
</a>
</th>
<th>
  <span th:text="${board.writer}">X</span>
</th>
<th>
  <span th:text="${#temporals.format(board.createdDate, 'yyyy-MM-dd
HH:mm')}">X</span>
</th>
</tr>

```



```

        <a sec:authorize="isAuthenticated()" class="btn btn-primary"
        th:href="@{/board/write}" role="button">글쓰기</a>
    </div>
</div>
<th:block th:replace="~/module/footer"></th:block>
</div>
</div>
<script src="/webjars/jquery/3.5.1/jquery.min.js"></script>
<script src="/webjars/bootstrap/4.5.0/js/bootstrap.min.js"></script>
</body>
</html>

```

7 게시판 글쓰기 페이지 작성 /resources/templates/board 폴더에 write.html 파일을 생성하여 다음의 내용을 작성합니다.

BookMarket/src/main/resources/templates/board/write.html

```

<html>
<head>
    <meta charset="UTF-8"/>
    <title>게시판 등록</title>
    <link href="/BookMarket/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
<div class="container py-4">
    <th:block th:replace="~/module/header"></th:block>
    <div class="p-5 mb-4 bg-body-tertiary rounded-3">
        <div class="container-fluid py-5">
            <h1 class="display-5 fw-bold">게시판 등록</h1>
            <p class="col-md-8 fs-4">BookMarket</p>
        </div>
    </div>
    <div class="row align-items-md-stretch">
        <form action="/BookMarket/board/write" method="post">
            <input type="hidden" name="writerid" class="form-control"
            th:value="${session.userLoginInfo.memberId}"/>

```

```

<div class="mb-3 row">
  <label class="col-sm-2 control-label">제목</label>
  <div class="col-sm-3">
    <input type="text" name="title" class="form-control"/>
  </div>
</div>
<div class="mb-3 row">
  <label class="col-sm-2 control-label">작성자</label>
  <div class="col-sm-3">
    <input type="text" name="writer" class="form-control"
      th:value="${session.userLoginInfo.name}"/>
  </div>
</div>
<div class="mb-3 row">
  <label class="col-sm-2 control-label">내용</label>
  <div class="col-sm-5">
    <textarea type="text" name="content" cols="50" rows="3" class="form-
      control">X</textarea>
  </div>
</div>
<div class="mb-3 row">
  <div class="col-sm-offset-2 col-sm-10">
    <input class="btn btn-primary" type="submit" role="button" value="글
      쓰기"/>
    <a href="/BookMarket/board/list" class="btn btn-secondary">취소</a>
  </div>
</div>
</form>
</div>
<th:block th:replace="~/module/footer">X</th:block>
</div>
</body>
</html>

```

8 게시물 상세 보기 페이지 작성 /resources/templates/board 폴더에 view.html 파일을 생성하

여 다음의 내용을 작성합니다.

BookMarket/src/main/resources/templates/board/view.html

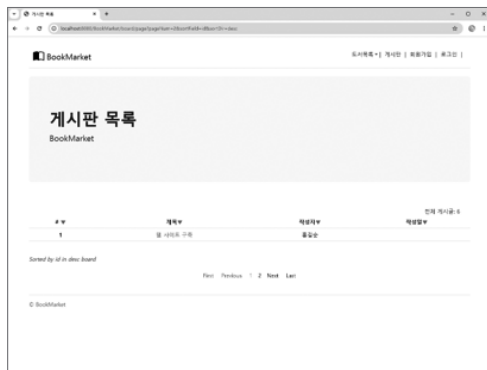
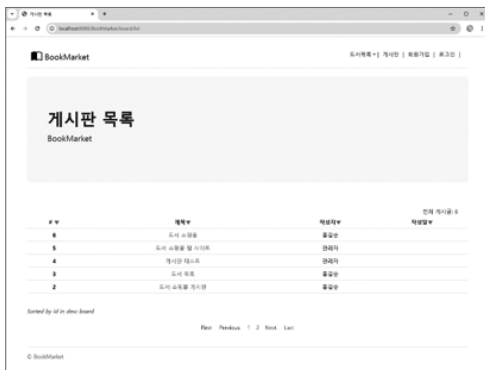
```
<html>
<head>
  <meta charset="UTF-8"/>
  <title>게시판 수정</title>
  <link href="/BookMarket/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
<div class="container py-4">
  <th:block th:replace="~/module/header"/></th:block>
  <div class="p-5 mb-4 bg-body-tertiary rounded-3">
    <div class="container-fluid py-5">
      <h1 class="display-5 fw-bold">게시판 수정</h1>
      <p class="col-md-8 fs-4">BookMarket</p>
    </div>
  </div>
  <div class="row align-items-md-stretch">
    <form th:object="${boardFormDto}" name="board" action="/BookMarket/board/
    update" method="post">
      <input type="hidden" name="id" class="form-control" th:field="*{id}"/>
      <input type="hidden" name="writerid" class="form-control"
      th:field="*{writerid}"/>
      <div class="mb-3 row">
        <label class="col-sm-2 control-label">제목</label>
        <div class="col-sm-3">
          <input type="text" name="title" class="form-control"
          th:field="*{title}"/>
        </div>
      </div>
      <div class="mb-3 row">
        <label class="col-sm-2 control-label">작성자</label>
        <div class="col-sm-3">
          <input type="text" name="writer" class="form-control"
          th:field="*{writer}"/>
        </div>
      </div>
    </form>
  </div>
</div>
```

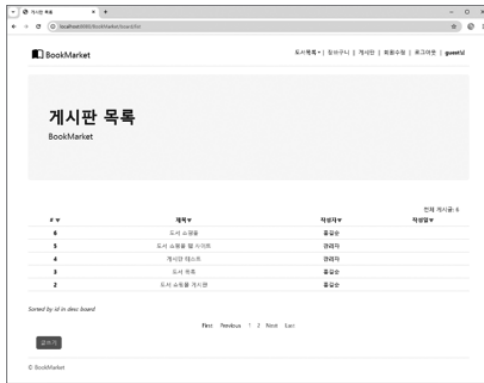
```

</div>
<div class="mb-3 row">
  <label class="col-sm-2 control-label">내용</label>
  <div class="col-sm-5">
    <textarea type="text" name="content" cols="50" rows="3" class="form-
      control" th:field="*{content}">/textarea>
  </div>
</div>
</div>
<div class="mb-3 row">
  <div class="col-sm-offset-2 col-sm-10">
    <a href="/BookMarket/board/list" class="btn btn-primary">목록</a>
    <input th:if="{buttonOk==true}" type="submit" class="btn btn-
      success" value="수정"/>
    <a th:if="{buttonOk==true}" th:href="@{'/board/delete/' + {id}}"
      class="btn btn-danger">삭제</a>
  </div>
</div>
</form>
</div>
<th:block th:replace="~/module/footer">/th:block>
</div>
</body>
</html>

```

9 프로젝트 실행 웹 브라우저에 `http://localhost:8080/BookMarket/`을 입력하여 실행합니다. 게시판 목록을 확인하거나 게시판 수정 등을 실행해 봅니다.



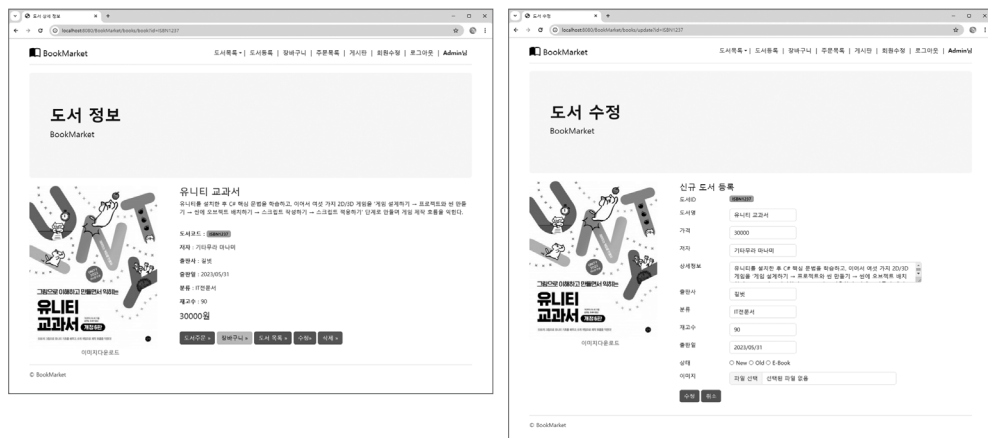


15.4

[도서쇼핑몰] 도서 CRUD 처리

스프링 데이터 JDBC로 데이터베이스와 연동하여 도서 쇼핑몰의 도서 목록 CRUD를 구현해 봅니다.

[미리보기] 도서 쇼핑몰의 도서 CRUD



실습 1 데이터베이스와 테이블 생성 및 데이터 등록하기

도서 쇼핑몰에서 메모리 저장소 객체에 저장한 전체 도서 목록을 데이터베이스에 저장하려고 합니다. 이를 위해서 먼저, 사용할 데이터베이스를 생성하고 도서 목록을 저장할 테이블을 생성하며 마지막으로 전체 도서 목록을 생성된 테이블에 삽입해 봅니다.

1 테이블 생성 및 도서 정보 삽입 /resources/statics/ 폴더에 sql 폴더를 만듭니다. 이 폴더에 book.sql 파일을 생성하여 다음 내용을 작성합니다.

USE BOOKMARKETDB; ❶

```
CREATE TABLE IF NOT EXISTS book (
    b_bookId VARCHAR(10) NOT NULL,
    b_name VARCHAR(30),
    b_unitPrice INTEGER,
    b_author VARCHAR(50),
    b_description TEXT,
    b_publisher VARCHAR(20),
    b_category VARCHAR(20),
    b_unitsInStock LONG,
    b_releaseDate VARCHAR(20),
    b_condition VARCHAR(20),
    b_fileName VARCHAR(20),
    PRIMARY KEY (b_bookId)
) DEFAULT CHARSET=utf8;
```

DELETE FROM book; ❸

```
INSERT INTO book VALUES('ISBN1234', '자바스크립트 입문', 30000, '조현영', '자바스크립트
의 기초부터 심화까지 핵심 문법을 학습한 후 12가지 프로그램을 만들어 학습한 내용을
확인할 수 있습니다. 문법 학습과 실습이 적절히 섞여 있어 프로그램을 만드는 방법을 재
미있게 익힐 수 있습니다.', '길벗', 'IT전문서', 1000, '2024/02/20', 'new', 'ISBN1234.
png');

INSERT INTO book VALUES('ISBN1235', '파이썬의 정석', 29800, '조용주, 임좌상', '4차 산업혁
명의 핵심인 머신러닝, 사물 인터넷(IoT), 데이터 분석 등 다양한 분야에 활용되는 직관
적이고 간결한 문법의 파이썬 프로그래밍 언어를 최신 트렌드에 맞게 예제 중심으로 학습
할 수 있습니다.', '길벗', 'IT교육교재', 1000, '2023/01/10', 'new', 'ISBN1235.png');

INSERT INTO book VALUES('ISBN1236', '안드로이드 프로그래밍', 25000, '송미영', '안드로
이드의 기본 개념을 체계적으로 익히고, 이를 실습 예제를 통해 익힙니다. 기본 개념
과 사용법을 스스로 실전에 적용하는 방법을 학습한 다음 실습 예제와 응용 예제를 통
해 실전 프로젝트 응용력을 키웁니다.', '길벗', 'IT교육교재', 1000, '2023/06/30',
'new', 'ISBN1236.png');
```

- ❶ 사용할 데이터베이스를 BOOKMARKETDB로 설정합니다.
- ❷ b_bookId(도서 ID), b_name(도서명), b_unitPrice(가격), b_author(저자), b_description(상세 정보), b_publisher(출판사), b_category(분류), b_unitsInStock(재고 수), b_totalPages(페이지 수), b_releaseDate(출판일), b_condition(상태), b_fileName(파일) 필드를 가진 book 테이블을 생성합니다.

- 3 book 테이블이 있다면 삭제합니다.
- 4 INSERT 구문으로 도서 목록을 저장합니다.

2 sql 파일 실행 /resources/static/sql/ 폴더에 있는 book.sql 파일의 내용을 불러와 SQL문으로 실행합니다. book 테이블을 생성하고 데이터를 일괄 삽입합니다.

```
...
mysql> source book.sql

Database changed
```

source 명령어는 MySQL에서 외부 파일의 SQL문을 실행시킵니다. source 명령어를 사용하는 방법은 다음과 같습니다.

```
mysql> source 파일이름;
```

source book.sql은 앞에서 생성한 book.sql 파일을 실행하여 테이블을 생성하고, 생성된 테이블에 레코드 세 개를 삽입하는 코드입니다. 여기에서 book.sql 파일은 MySQL을 접속하기 전 실행한 경로와 반드시 같아야 합니다.

접속된 MySQL의 현재 경로와 book.sql 파일이 위치한 경로가 다르다면 다음과 같이 book.sql 파일이 위치한 경로 이름을 정확히 표시해야 합니다.

```
mysql> source C:/프로젝트경로/WebContent/resources/sql/book.sql
```

또는 book.sql을 복사한 후 Mysql 접속 화면에 붙여넣기를 한 후 **[Enter]**를 누릅니다.

```
MySQL 5.6 Command Line Client
mysql> USE BOOKMANHETD;
Database changed
mysql>
mysql> CREATE TABLE IF NOT EXISTS book(
  ->   b_bookid VARCHAR(18) NOT NULL,
  ->   b_name VARCHAR(30),
  ->   b_unitprice INT(8),
  ->   b_author VARCHAR(50),
  ->   b_publisher VARCHAR(20),
  ->   b_category VARCHAR(20),
  ->   b_unitinstock LONG,
  ->   b_releasedate VARCHAR(20),
  ->   b_condition VARCHAR(20),
  ->   b_filename VARCHAR(20),
  ->   PRIMARY KEY (b_bookid)
  -> )DEFAULT CHARSET=utf8;
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql> DELETE FROM book;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO book VALUES('ISBN1234', '자비스크림트 인공', 30000, '조현민', '자비스크림트 인공의 기초부터 심화까지 핵심 공법을 학습한 12가지 프로그램을 일괄적 학습한 내용물 확인할 수 있습니다. 문법 학습과 실습이 적절히 섞여 있어 프로그램을 만드는 방법을 체계있게 익힐 수 있습니다.', '공통', 'IT전문서', 1000, '2023/02/28', 'new', 'ISBN1234.png');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO book VALUES('ISBN1235', '자이언트 정석', 20000, '조용주, 임희상', '4차 산업혁명의 핵심인 인공지능, 사물 인터넷(IoT), 데이터 분석 등 다양한 분야의 활용되는 인공지능 기술의 최신 트렌드와 실제 사례를 중심으로 학습할 수 있습니다.', '공통', 'IT교육교재', 1000, '2023/03/30', 'new', 'ISBN1235.png');
Query OK, 1 row affected (0.00 sec)

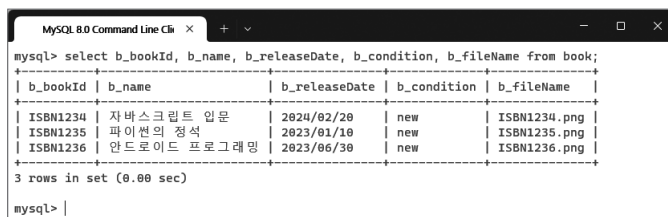
mysql> INSERT INTO book VALUES('ISBN1236', '안드로이드 프로그래밍', 25000, '송의영', '안드로이드의 기본 개념을 체계적으로 익히고, 여러 실습 예제를 통해 익힙니다. 기본 개념과 사용법을 스스로 실전에 적용하는 방법을 학습할 수 있습니다. 실습 예제와 응용 예제를 통해 실전 프로젝트 응용력을 기릅니다.', '공통', 'IT교육교재', 1000, '2023/06/30', 'new', 'ISBN1236.png');
Query OK, 1 row affected (0.00 sec)
```

3 book 테이블 확인 삽입된 데이터를 확인합니다.

```
mysql> SELECT b_bookId, b_name, b_releaseDate, b_condition, b_fileName FROM book;
```

이 코드는 book 테이블에서 b_bookId, b_name, b_releaseDate, b_condition, b_fileName 값만 조회합니다. book 테이블의 모든 필드에 등록된 레코드를 조회할 때는 select * from book으로 입력해야 합니다.

다음은 테이블 이름 book의 모든 필드 이름과 레코드 값을 출력한 결과입니다.



```
mysql> select b_bookId, b_name, b_releaseDate, b_condition, b_fileName from book;
```

b_bookId	b_name	b_releaseDate	b_condition	b_fileName
ISBN1234	자바스크립트 입문	2024/02/20	new	ISBN1234.png
ISBN1235	파이썬의 정석	2023/01/10	new	ISBN1235.png
ISBN1236	안드로이드 프로그래밍	2023/06/30	new	ISBN1236.png

3 rows in set (0.00 sec)

```
mysql>
```

실습 2 JDBC 연동을 위한 환경 설정하기

도서 쇼핑몰 애플리케이션에 데이터베이스를 연동하기 위해 환경 설정을 합니다.

1 의존 라이브러리 등록 build.gradle 파일에 JDBC로 데이터베이스 연동과 관련된 의존 라이브러리를 추가하고 재실행합니다.

```
build.gradle

dependencies {
    ...
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    implementation 'org.springframework.boot:spring-boot-starter-jdbc'
    implementation group: 'com.mysql', name: 'mysql-connector-j', version: '8.0.33'
}
```

2 데이터베이스 관련 환경 설정 application.properties 파일을 다음과 같이 추가 작성합니다.

```
BookMarket/src/main/resources/application.properties

spring.application.name=BookMarket
server.servlet.context-path=/BookMarket
...
spring.datasource.url=jdbc:mysql://localhost:3306/bookmarketDB // 데이터베이스 URL
spring.datasource.username=root // 관리자 계정 ID
spring.datasource.password=1234 // 관리자 비밀번호
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver // 드라이버 클래스 이름

spring.jpa.database=mysql // 데이터베이스 종류
spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect // 데이터베이스 플랫폼
spring.jpa.hibernate.ddl-auto=update // 엔티티의 변경된 부분만 적용
spring.jpa.generate-ddl=false // 시작 시 스키마 초기화하지 않음
spring.jpa.show-sql=true // SQL문 로깅 활성화함
spring.jpa.properties.hibernate.format_sql=true // 로그, 콘솔에 SQL문을 출력함
```

1 데이터베이스와 관련된 스프링 데이터 소스를 설정합니다.

2 스프링 데이터 JPA를 설정합니다.

실습 3 도서 목록 조회하기

JdbcTemplate 클래스의 조회 메서드로 도서 쇼핑몰의 데이터베이스에 저장된 전체 도서 목록을 조회하여 가져온 뒤 뷰 페이지에 출력하도록 구현해 보니다

1 도서 조회 결과 가져오기 com.springboot.repository 패키지에 BookRowMapper.java 파일을 생성하여 다음의 내용을 작성합니다.

```
BookMarket/src/main/java/com/springboot/repository/BookRowMapper.java

package com.springboot.repository;
import java.math.BigDecimal;
import java.sql.ResultSet;
```

```

import java.sql.SQLException;
import org.springframework.jdbc.core.RowMapper;
import com.springboot.domain.Book;
import lombok.Data;

@Data
public class BookRowMapper implements RowMapper<Book> {
    public Book mapRow(ResultSet rs, int rowNum) throws SQLException {
        Book book = new Book();
        book.setBookId(rs.getString(1));
        book.setName(rs.getString(2));
        book.setUnitPrice(new BigDecimal(rs.getInt(3)));
        book.setAuthor(rs.getString(4));
        book.setDescription(rs.getString(5));
        book.setPublisher(rs.getString(6));
        book.setCategory(rs.getString(7));
        book.setUnitsInStock(rs.getLong(8));
        book.setReleaseDate(rs.getString(9));
        book.setCondition(rs.getString(10));
        book.setFileName(rs.getString(11));
        return book;
    }
}

```

① 데이터베이스에 등록된 도서 목록의 Book 객체를 목록에 담기 위해 BookRowMapper 클래스를 생성합니다.

② 전체 도서 목록 가져오기 com.springboot.repository 패키지의 BookRepositoryImpl 클래스에 다음 내용을 수정합니다.

BookMarket/src/main/java/com/springboot/repository/BookRepositoryImpl.java

```

package com.springboot.repository;

...

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.BeanPropertyRowMapper;
import org.springframework.jdbc.core.JdbcTemplate;

```

```

@Repository
public class BookRepositoryImpl implements BookRepository {

    @Autowired
    private JdbcTemplate jdbcTemplate;

    public List<Book> getAllBookList() {
        String sql = "SELECT * FROM book";
        List<Book> listOfBooks = jdbcTemplate.query(sql, , new BookRowMapper());
        return listOfBooks;
    }
    ...
}

```

- ❶ 데이터베이스의 book 테이블에 등록된 전체 도서 목록을 조회하고 반환하는 메서드입니다.
- ❷ SQL문을 SELECT * FROM book으로 간단히 설정합니다.
- ❸ 도서 목록 조회이므로 query() 메서드를 사용합니다. queryForList() 메서드를 대신 사용할 수도 있는데 이 메서드는 RowMapper를 제공하지 않으므로 다음과 같이 변경해야 합니다.

```

public List<Book> getAllBookList() {
    String sql = "SELECT * FROM book";
    List<Book> listOfBooks = new ArrayList<>();
    List<Map<String, Object>> rows = this.jdbcTemplate.queryForList(sql);
    for (Map<String, Object> row : rows) {
        Book book = new Book();
        book.setBookId((String)row.get("b_bookId"));
        book.setName((String)row.get("b_name"));
        book.setUnitPrice(new BigDecimal((Integer)row.get("b_unitPrice")));
        book.setAuthor((String)row.get("b_author"));
        book.setDescription((String)row.get("b_description"));
        book.setPublisher((String)row.get("b_publisher"));
        book.setCategory((String)row.get("b_category"));
        book.setUnitsInStock(new Long((String)row.get("b_unitsInStock")));
        book.setReleaseDate((String)row.get("b_releaseDate"));
        book.setCondition((String)row.get("b_condition"));
        book.setFileName((String)row.get("b_fileName"));
    }
    return listOfBooks;
}

```

```

        listOfBooks.add(book);
    }
    return listOfBooks;
}

```

3 도서 ID 정보 가져오기 com.springboot.repository 패키지의 BookRepositoryImpl 클래스에 다음 내용을 수정합니다.

15

```

BookMarket/src/main/java/com/springboot/repository/BookRepositoryImpl.java

package com.springboot.repository;
...
@Repository
public class BookRepositoryImpl implements BookRepository {
    ...
    public Book getBookById(String bookId) {
        Book bookInfo = null;
        String sql = "SELECT count(*) FROM book where b_bookId=?";
        int rowCount = jdbcTemplate.queryForObject(sql, Integer.class, bookId);
        if (rowCount != 0) {
            sql = "SELECT * FROM book where b_bookId=?";
            bookInfo = jdbcTemplate.queryForObject(sql, new BookRowMapper(), bookId);
        }
        if (bookInfo == null)
            throw new BookIdException(bookId);
        return bookInfo;
    }
    ...
}

```

- ① 데이터베이스의 book 테이블에 등록된 전체 도서 목록 중에서 검색 조건인 도서 ID와 일치하는 도서를 조회하여 해당 도서를 반환하는 메서드입니다.
- ② 등록된 전체 도서 목록 중에서 검색 조건인 도서 ID와 일치하는 레코드 개수를 얻어 옵니다. 이것은 검색 조건인 도서 ID가 등록될 때만 데이터베이스에 접근하여 해당 도서를 조회할 수 있도록 하기 위함입니다.

- ③ 레코드 개수가 한 개 이상일 때 검색 조건인 도서 ID와 일치하는 도서를 조회합니다.
- ④ SQL문을 `SELECT * FROM book WHERE b_bookId=?`로 간단히 설정합니다.
- ⑤ 도서 ID는 book 테이블의 기본 키로 검색 조건인 도서 ID에 대한 도서는 한 개만 있으므로 `queryForObject()` 메서드를 사용합니다.

④ **도서 분류 가져오기** `com.springboot.repository` 패키지의 `BookRepositoryImpl` 클래스에 다음 내용을 수정합니다.

```

BookMarket/src/main/java/com/springboot/repository/BookRepositoryImpl.java

package com.springboot.repository;
...
@Repository
public class BookRepositoryImpl implements BookRepository {
    ...
    public List<Book> getBookListByCategory(String category) {
        List<Book> booksByCategory = new ArrayList<Book>();
        String sql = "SELECT * FROM book where b_category LIKE '%" + category + "%'";
        booksByCategory = jdbcTemplate.query(sql, new BookRowMapper());
        return booksByCategory;
    }
    ...
}

```

- ① 데이터베이스의 book 테이블에 등록된 모든 도서 목록 중에서 검색 조건인 도서 분류(category)와 일치하는 전체 도서 목록을 조회하여 반환하는 메서드입니다.
- ② SQL문을 `SELECT * FROM book WHERE b_category LIKE '%" + category + "%'`로 간단히 설정해도 됩니다.
- ③ 검색 조건인 도서 분류에 대한 도서 목록을 조회하므로 `query()` 메서드를 `RowMapper`로 변경합니다. 또는 `queryForList()` 메서드를 사용할 수도 있습니다.

```

public List<Book> getBookListByCategory(String category) {
    List<Book> booksByCategory = new ArrayList<Book>();
    String sql = "SELECT * FROM book where b_category LIKE '%" + category + "%'";
    List<Map<String, Object>> rows = this.jdbcTemplate.queryForList(sql);
}

```



```

for (Map<String, Object> row : rows) {
    Book book = new Book();
    book.setBookId((String)row.get("b_bookId"));
    book.setName((String)row.get("b_name"));
    book.setUnitPrice(new BigDecimal((Integer)row.get("b_unitPrice")));
    book.setAuthor((String)row.get("b_author"));
    book.setDescription((String)row.get("b_description"));
    book.setPublisher((String)row.get("b_publisher"));
    book.setCategory((String)row.get("b_category"));
    book.setUnitsInStock(new Long((String)row.get("b_unitsInStock")));
    book.setReleaseDate((String)row.get("b_releaseDate"));
    book.setCondition((String)row.get("b_condition"));
    book.setFileName((String)row.get("b_fileName"));
    booksByCategory.add(book);
}
return booksByCategory;
}

```

5 도서 필터 정보 가져오기 com.springboot.repository 패키지의 BookRepositoryImpl 클래스에 다음 내용을 수정합니다.

BookMarket/src/main/java/com/springboot/repository/BookRepositoryImpl.java

```

package com.springboot.repository;
...
@Repository
public class BookRepositoryImpl implements BookRepository {
    ...
    public Set<Book> getBookListByFilter(Map<String, List<String>> filter) {
        Set<Book> booksByPublisher = new HashSet<Book>();
        Set<Book> booksByCategory = new HashSet<Book>();
        Set<String> booksByFilter = filter.keySet();
        if (booksByFilter.contains("publisher")) {
            for (int j = 0; j < filter.get("publisher").size(); j++) {
                String publisherName = filter.get("publisher").get(j);
            }
        }
    }
}

```

```

        String sql = "SELECT * FROM book where b_publisher LIKE '%" +
            publisherName + "%'"; ❷

        List<Book> book = jdbcTemplate.query(sql, BeanPropertyRowMapper.
            newInstance(Book.class));
        booksByPublisher.addAll(book);
    }
}

if (booksByFilter.contains("category")) {
    for (int i = 0; i < filter.get("category").size(); i++) {
        String category = filter.get("category").get(i);
        String sql = "SELECT * FROM book where b_category LIKE '%" + category +
            "%'"; ❹

        List<Book> list = jdbcTemplate.query(sql, BeanPropertyRowMapper.
            newInstance(Book.class));
        booksByCategory.addAll(list);
    }
}

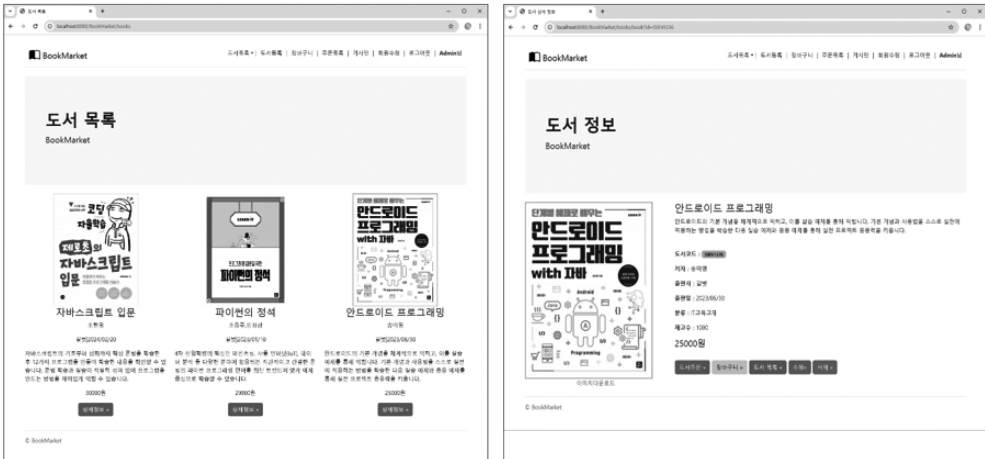
booksByCategory.retainAll(booksByPublisher);
return booksByCategory;
}

...
}

```

- ❶ 데이터베이스의 book 테이블에 등록된 모든 도서 목록 중에서 검색 조건인 출판사 (publisher) 및 도서 분류(category)와 일치하는 전체 도서 목록을 조회하여 반환하는 메서드입니다.
- ❷ SQL문을 SELECT * FROM book WHERE b_publisher LIKE '%' + publisherName + '%'로 간단히 설정합니다.
- ❸ 검색 조건인 출판사에 대한 도서 목록을 조회하므로 query() 메서드를 사용합니다. 또는 queryForList() 메서드를 사용할 수도 있습니다.
- ❹ SQL문을 SELECT * FROM book WHERE b_category LIKE '%' + category + '%'로 설정합니다.
- ❺ 검색 조건인 도서 분류에 대한 도서 목록을 조회하므로 query() 메서드를 사용합니다. 또는 queryForList() 메서드를 사용할 수도 있습니다.

6 프로젝트 실행 웹 브라우저에 `http://localhost:8080/BookMarket/books`를 입력하여 실행 결과를 확인합니다.



실습 4 신규 도서 삽입하기

JdbcTemplate 클래스의 `update()` 메서드를 이용하여 도서 쇼핑몰에서 새로운 도서 정보를 저장하고 뷰 페이지에 출력하는 것을 구현해 봅니다.

1 신규 도서 삽입 `com.springboot.repository` 패키지의 `BookRepositoryImpl` 클래스에 다음 내용을 수정합니다.

BookMarket/src/main/java/com/springboot/repository/BookRepositoryImpl.java

```
package com.springboot.repository;

...

@Repository
public class BookRepositoryImpl implements BookRepository {

    ...

    public void setNewBook(Book book) { // book 테이블에 신규 도서를 저장하는 메서드
        String SQL = "INSERT INTO book (b_bookId, b_name, b_unitPrice, b_author, b_
            description, b_publisher, b_category, b_unitsInStock, b_releaseDate, b_
            condition, b_fileName)" + "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
        jdbcTemplate.update(SQL, book.getBookId(), book.getName(), book.getUnitPrice(),
            book.getAuthor(), book.getDescription(), book.getPublisher(),
```

```

        book.getCategory(), book.getUnitsInStock(), book.
        getReleaseDate(), book.getCondition(), book.getFileName());
    }
}

```



실습 5 도서 정보 수정하기

JdbcTemplate 클래스의 update() 메서드를 이용하여 도서 쇼핑몰에서 도서 정보를 수정하고 뷰 페이지에 출력하는 것을 구현해 봅니다.

1 저장소 인터페이스 수정 com.springboot.repository 패키지의 BookRepository 인터페이스에 다음 내용을 추가 작성합니다.

BookMarket/src/main/java/com/springboot/repository/BookRepository.java

```

package com.springboot.repository;

...

public interface BookRepository {

    ...

    void setNewBook(Book book);

    void setUpdateBook(Book book);

}

```

2 저장소 클래스 수정 com.springboot.repository 패키지의 BookRepositoryImpl 클래스에 다음 내용을 추가 작성합니다.

```
BookMarket/src/main/java/com/springboot/repository/BookRepositoryImpl.java

package com.springboot.repository;

...
@Repository
public class BookRepositoryImpl implements BookRepository {

    ...
    public void setUpdateBook(Book book) {
        if (book.getFileName() != null) {
            String SQL = "UPDATE Book SET b_name = ?, b_unitPrice = ?, b_author = ?, b_
                description = ?, b_publisher = ?, b_category = ?, b_unitsInStock =
                ?, b_releaseDate = ?, b_condition = ?, b_fileName = ? where b_bookId
                = ? ";
            jdbcTemplate.update(SQL, book.getName(), book.getUnitPrice(), book.
                getAuthor(), book.getDescription(), book.getPublisher(),
                book.getCategory(), book.getUnitsInStock(), book.
                getReleaseDate(), book.getCondition(), book.getFileName(),
                book.getBookId());
        } else if (book.getFileName() == null) {
            String SQL = "UPDATE Book SET b_name = ?, b_unitPrice = ?, b_author = ?, b_
                description = ?, b_publisher = ?, b_category = ?, b_unitsInStock = ?,
                b_releaseDate = ?, b_condition = ? where b_bookId = ? ";
            jdbcTemplate.update(SQL, book.getName(), book.getUnitPrice(), book.
                getAuthor(), book.getDescription(), book.getPublisher(),
                book.getCategory(), book.getUnitsInStock(), book.
                getReleaseDate(), book.getCondition(), book.getBookId());
        }
    }
}
```

① 정보 수정을 요청한 도서의 내용을 갱신하는 메서드입니다. 업로드하는 도서 이미지가 없
을 때는 b_fileName(도서의 이미지 파일 이름) 필드를 제외한 도서 내용을 갱신합니다.

3 서비스 인터페이스 수정 com.springboot.service 패키지의 BookService 인터페이스에 다음
내용을 추가 작성합니다.

```

BookMarket/src/main/java/com/springboot/service/BookService.java

package com.springboot.service;
...
public interface BookService {
    ...
    void setNewBook(Book book);
    void setUpdateBook(Book book);
}

```

4 서비스 클래스 수정 com.springboot.servive 패키지의 BookServiceImpl 클래스에 다음 내용을 추가 작성합니다.

```

BookMarket/src/main/java/com/springboot/service/BookServiceImpl.java

package com.springboot.service;
...
@Service
public class BookServiceImpl implements BookService {
    ...
    public void setNewBook(Book book) {
        bookRepository.setNewBook(book);
    }
    public void setUpdateBook(Book book) { // 정보 수정 요청한 도서 내용을 갱신하는 메서드
        bookRepository.setUpdateBook(book);
    }
}

```

5 도서 정보 수정 com.springboot.controller 패키지의 BookController 클래스에 다음 내용을 추가 작성합니다.

```

BookMarket/src/main/java/com/springboot/controller/BookController.java

package com.springboot.controller;
...
@Controller

```

```

@RequestMapping("/books")
public class BookController {

    ...

    @GetMapping("/update")
    public String getUpdateBookForm(@ModelAttribute("updateBook") Book book,
        @RequestParam("id") String bookId, Model model) {
        Book bookById = bookService.getBookById(bookId);
        model.addAttribute("book", bookById);
        return "updateForm";
    }

    @PostMapping("/update")
    public String processUpdateBookForm(@ModelAttribute("updateBook") Book book) {
        MultipartFile bookImage = book.getBookImage();
        String rootDirectory = fileDir;
        if (bookImage != null && !bookImage.isEmpty()) {
            try {
                String fname = bookImage.getOriginalFilename();
                bookImage.transferTo(new File(fileDir + fname));
                book.setFileName(fname);
            } catch (Exception e) {
                throw new RuntimeException("Book Image saving failed", e);
            }
        }
        bookService.setUpdateBook(book);
        return "redirect:/books";
    }
}

```

- ❶ getUpdateBookForm()은 웹 브라우저에서 사용자의 요청 URL이 `http://.../books/update`고, HTTP 메서드가 GET 방식이면 매핑되는 요청 처리 메서드입니다. 수정하려는 도서 정보를 updateBook 커맨드 객체로 뷰 페이지에 전달하고 뷰 이름인 updateForm을 반환합니다. 웹 브라우저의 뷰 페이지는 JSP 파일인 updateForm.jsp가 출력됩니다.
- ❷ submitUpdateBookForm()은 웹 브라우저에서 사용자의 요청 URL이 `http://.../books/update`고, HTTP 메서드가 POST 방식이면 매핑되는 요청 처리 메서드입니다. 뷰 페이지인 updateForm.jsp에서 <form:input> 태그에 입력된 수정 데이터 값은 updateBook 커맨드 객체로 전달되며, 웹 요청 처리 메서드인 submitUpdateBookForm()에서 전달받은

updateBook 커맨드 객체의 데이터 값으로 도서 정보가 수정됩니다. 이때 수정 요청을 처리한 후 URL은 `http://.../books`로 이동합니다.

6 도서 상세 정보 페이지 수정 book.html 파일을 다음과 같이 수정합니다.

```
BookMarket/src/main/resources/templates/book.html

<html xmlns:sec="http://www.thymeleaf.org/extras/spring-security">
<head>
...
    <form name="addForm" id="addForm" method="post">
        <input type="hidden" name="_method" value="put"/>
        <a href="/BookMarket/cart" class="btn btn-warning">장바구니 &raquo;</a>
        <a href="/BookMarket/books" class="btn btn-secondary">도서 목록 &raquo;</a>
        <a sec:authorize="hasRole('ROLE_ADMIN')" th:href="/BookMarket/books/
            update?id='${book.bookId}" class="btn btn-success">수정&raquo;</a>
    </form>
...
```

- 1 뷰 페이지에서 시큐리티 태그를 사용하기 위해 태그 라이브러리를 선언합니다.
- 2 `sec:authorize="hasRole('ROLE_ADMIN')"`은 관리자가 인증된 경우에 [수정] 버튼을 표시합니다.

7 도서 수정 페이지 작성 updateForm.html을 생성하고 다음의 내용을 작성합니다.

```
BookMarket/src/main/resources/templates/updateForm.html

<html>
<head>
    <meta charset="UTF-8">
    <title>도서 수정</title>
    <link href="/BookMarket/css/bootstrap.min.css" rel="stylesheet">
    <script src="/BookMarket/js/controllers.js"></script>
</head>
<body>
    <div class="container py-4">
        <header class="pb-3 mb-4 border-bottom">
```



```

<a href="/BookMarket/home" class="d-flex align-items-center text-body-emphasis
text-decoration-none">
    <svg xmlns="http://www.w3.org/2000/svg" width="32" height="32"
    fill="currentColor" class="bi bi-book-half me-2" viewBox="0 0 16 16">
        <path d="M8.5 2.687c.654-.689 1.782-.886 3.112-.752 1.234.124 2.503.523
        3.388.893v9.923c-.918-.35-2.107-.692-3.287-.81-1.094-.111-2.278-
        .039-3.213.492zM8 1.783C7.015.936 5.587.81 4.287.94c-1.514.153-
        3.042.672-3.994 1.105A.5.5 0 0 0 2.5 1.1a.5.5 0 0 0 .707.455c.882-
        .4 2.303-.881 3.68-1.02 1.409-.142 2.59.087 3.223.877a.5.5 0 0 0 .78
        0c.633-.79 1.814-1.019 3.222-.877 1.378.139 2.8.62 3.681 1.02A.5.5
        0 0 0 16 13.5v-11a.5.5 0 0 0-.293-.455c-.952-.433-2.48-.952-3.994-
        1.105C10.413.809 8.985.936 8 1.783"/>
    </svg>
    <span class="fs-4">BookMarket</span>
</a>
</header>
<div class="p-5 mb-4 bg-body-tertiary rounded-3">
    <div class="container-fluid py-5">
        <h1 class="display-5 fw-bold">도서 수정</h1>
        <p class="col-md-8 fs-4">BookMarket</p>
    </div>
</div>
<div class="row align-items-md-stretch">
    <div class="col-md-4">
        
        <div class="text-center">
            <a th:href="@{download(file=${book.fileName})}" style="text-decoration-
            line:none">이미지 다운로드</a>
        </div>
    </div>
    <div class="col-md-8">
        <form th:object="${book}" action="/BookMarket/books/update" method="post"
        enctype="multipart/form-data">
            <div class="mb-3 row">
                <label class="col-sm-2 control-label">도서 ID</label>
                <div class="col-sm-3">

```

```

        <input type="hidden" name="bookId" class="form-control"
            th:field="*{bookId}" readonly="readonly"/>
        <span class="badge text-bg-info" th:text ="${book.bookId}">X/
        span>
    </div>
</div>
<div class="mb-3 row">
    <label class="col-sm-2 control-label">도서명</label>
    <div class="col-sm-3">
        <input type="text" name="name" class="form-control"
            th:field="*{name}"/> ❶
    </div>
</div>
<div class="mb-3 row">
    <label class="col-sm-2 control-label">가격</label>
    <div class="col-sm-3">
        <input type="text" name="unitPrice" class="form-control"
            th:field="*{unitPrice}"/> ❶
    </div>
</div>
<div class="mb-3 row">
    <label class="col-sm-2 control-label">저자</label>
    <div class="col-sm-3">
        <input type="text" name="author" class="form-control"
            th:field="*{author}"/> ❶
    </div>
</div>
<div class="mb-3 row">
    <label class="col-sm-2 control-label">상세 정보</label>
    <div class="col-sm-8">
        <textarea type="text" name="description" cols="50" rows="2"
            class="form-control" th:field="*{description}">X</textarea> ❶
    </div>
</div>
<div class="mb-3 row">
    <label class="col-sm-2 control-label">출판사</label>
    <div class="col-sm-3">

```

```

        <input type="text" name="publisher" class="form-control"
            th:field="*{publisher}"/> ❶
    </div>
</div>
<div class="mb-3 row">
    <label class="col-sm-2 control-label">분류</label>
    <div class="col-sm-3">
        <input type="text" name="category" class="form-control"
            th:field="*{category}"/> ❶
    </div>
</div>
<div class="mb-3 row">
    <label class="col-sm-2 control-label">재고 수</label>
    <div class="col-sm-3">
        <input type="text" name="unitsInStock" class="form-control"
            th:field="*{unitsInStock}"/> ❶
    </div>
</div>
<div class="mb-3 row">
    <label class="col-sm-2 control-label">출판일</label>
    <div class="col-sm-3">
        <input type="text" name="releaseDate" class="form-control"
            th:field="*{releaseDate}"/> ❶
    </div>
</div>
<div class="mb-3 row">
    <label class="col-sm-2 control-label">상태</label>
    <div class="col-sm-3">
        <input type="radio" name="condition" value="New"/>New
        <input type="radio" name="condition" value="Old"/>Old
        <input type="radio" name="condition" value="E-Book"/>E-Book
    </div>
</div>
<div class="mb-3 row">
    <label class="col-sm-2 control-label">이미지</label>
    <div class="col-sm-7">
        <input type="file" name="bookImage" class="form-control"/>
    </div>
</div>

```

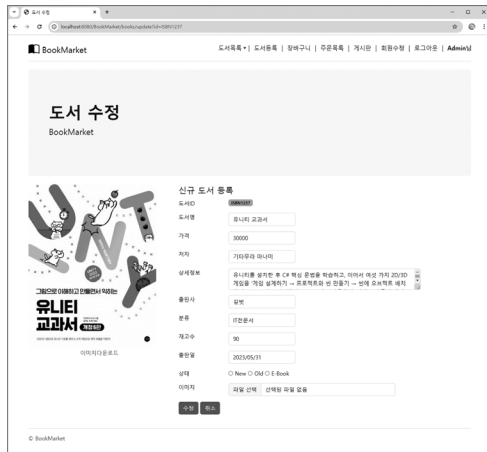
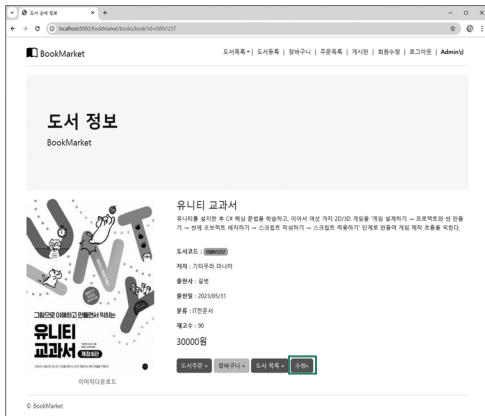
```

        </div>
    </div>
    <div class="mb-3 row">
        <div class="col-sm-offset-2 col-sm-10">
            <input type="submit" class="btn btn-primary" value ="수정"/>
            <a href="/BookMarket/books" class="btn btn-primary">취소</a>
        </div>
    </div>
</form>
</div>
<div>
    <div class="pt-3 mt-4 text-body-secondary border-top">
        <span class="text-body-secondary">&copy; BookMarket</span>
    </div>
</div>
</body>
</html>

```

① <input> 태그 내에 설정된 커맨드 객체의 필드 이름에 따라 필드 값을 출력합니다.

8 프로젝트 실행 웹 브라우저에 <http://localhost:8080/BookMarket/books>를 입력하여 실행한 후 [상세보기]>> 버튼을 눌러 도서 정보 화면으로 이동합니다. 이 화면에서 [수정] 버튼을 눌러 입력 항목을 수정한 뒤 실행 결과를 확인합니다.



실습 6 도서 삭제하기

JdbcTemplate 클래스의 update() 메서드를 이용하여 도서 쇼핑몰에서 도서 정보를 삭제하여 뷰 페이지에 출력하는 것을 구현합니다.

- 1 **저장소 인터페이스 수정** com.springboot.repository 패키지의 BookRepository 인터페이스에 다음 내용을 추가 작성합니다.

BookMarket/src/main/java/com/springboot/repository/BookRepository.java

```
package com.springboot.repository;

...

public interface BookRepository {

    ...

    void setUpdateBook(Book book);

    void setDeleteBook(String bookID);

}
```

- 2 **저장소 클래스 수정** com.springboot.repository 패키지의 BookRepositoryImpl 클래스에 다음 내용을 추가 작성합니다.

BookMarket/src/main/java/com/springboot/repository/BookRepositoryImpl.java

```
package com.springboot.repository;

...

@Repository
public class BookRepositoryImpl implements BookRepository {

    ...

    // 요청 도서 ID에 해당하는 도서를 데이터베이스에서 삭제하는 메서드
    public void setDeleteBook(String bookID) {

        String SQL = "DELETE from Book where b_bookId = ? ";

        jdbcTemplate.update(SQL, bookID);

    }

}
```

3 서비스 인터페이스 수정 com.springboot.service 패키지의 BookService 인터페이스 파일에 다음 내용을 추가 작성합니다.

```
BookMarket/src/main/java/com/springboot/service/BookService.java

package com.springboot.service;
...
public interface BookService {
    ...
    void setUpdateBook(Book book);
    void setDeleteBook(String bookID);
}
```

4 서비스 클래스 수정 com.springboot.service 패키지의 BookServiceImpl 클래스에 다음 내용을 추가 작성합니다.

```
BookMarket/src/main/java/com/springboot/service/BookServiceImpl.java

package com.springboot.service;
...
@Service
public class BookServiceImpl implements BookService {
    ...
    // 요청 도서 ID에 해당하는 도서를 데이터베이스에서 삭제하는 메서드
    public void setDeleteBook(String bookID) {
        bookRepository.setDeleteBook(bookID);
    }
}
```

5 도서 삭제 com.springboot.controller 패키지의 BookController 클래스에 다음 내용을 추가 작성합니다.

```
BookMarket/src/main/java/com/springboot/controller/BookController.java

package com.springboot.controller;
...
@Controller
```

```

@RequestMapping(value = "/books")
public class BookController {
    ...
    @RequestMapping(value = "/delete")
    public String getDeleteBookForm(Model model, @RequestParam("id") String bookId) {
        bookService.setDeleteBook(bookId);
        return "redirect:/books";
    }
}

```

- ❶ getDeleteBookForm()은 웹 브라우저에서 사용자의 요청 URL이 `http://.../books/delete`이고, HTTP 메서드가 GET 방식이면 매핑되는 요청 처리 메서드입니다. 이 메서드는 요청 도서 ID에 해당하는 도서를 데이터베이스에서 삭제합니다. 요청을 처리한 후 URL은 `http://.../books`로 이동합니다.

- ❹ 자바스크립트 수정 `resources/static/js/` 폴더의 `controller`에 다음 내용을 추가 작성합니다.

```

BookMarket/src/main/resources/static/js/controller.js
...
function deleteConfirm(id) {
    if (confirm("삭제 합니다!!") == true) location.href = "/BookMarket/books/delete?id="+id;
    else return;
}

```

- ❶ deleteConfirm() 메서드는 요청 도서를 삭제할 때 확인한 후 삭제하기 위한 자바스크립트입니다.

- ❺ 도서 상세 정보 페이지 수정 `book.html` 파일을 다음과 같이 추가 작성합니다.

```

BookMarket/src/main/resources/templates/book.html
<html xmlns:sec="http://www.thymeleaf.org/extras/spring-security">
...
<a sec:authorize="hasRole('ROLE_ADMIN')" th:href="/BookMarket/books/update?id='${book.bookId}'" class="btn btn-success">수정</a>

```

```

<a sec:authorize="hasRole('ROLE_ADMIN')" th:href="'javascript:deleteConfirm(
    ${book.bookId}+' class="btn btn-danger">삭제 &raquo;</a> ❶
</form>

...

```

❶ [삭제] 버튼을 누르면 해당 도서를 삭제하기 위해 자바스크립트 deleteConfirm() 메서드를 호출하여 도서 ID를 전달합니다.

❸ **프로젝트 실행** 웹 브라우저에 `http://localhost:8080/BookMarket/books`를 입력하여 실행한 후 [상세보기>>] 버튼을 눌러 도서 정보 화면으로 이동합니다. 이 화면에서 [삭제] 버튼을 눌러 실행 결과를 확인합니다.

