20. 스프링 프로그래밍 응용

스프링 프로그래밍 응용

날짜 : 2023/ 05/ 16 (화)
이름 : 김재준

π

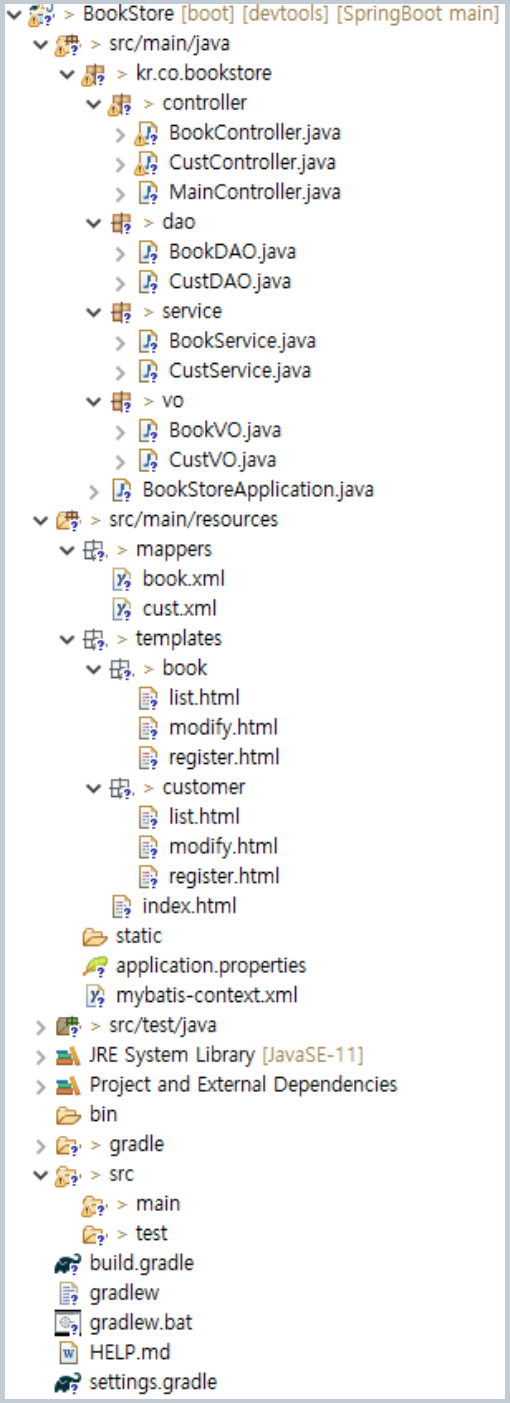
목차

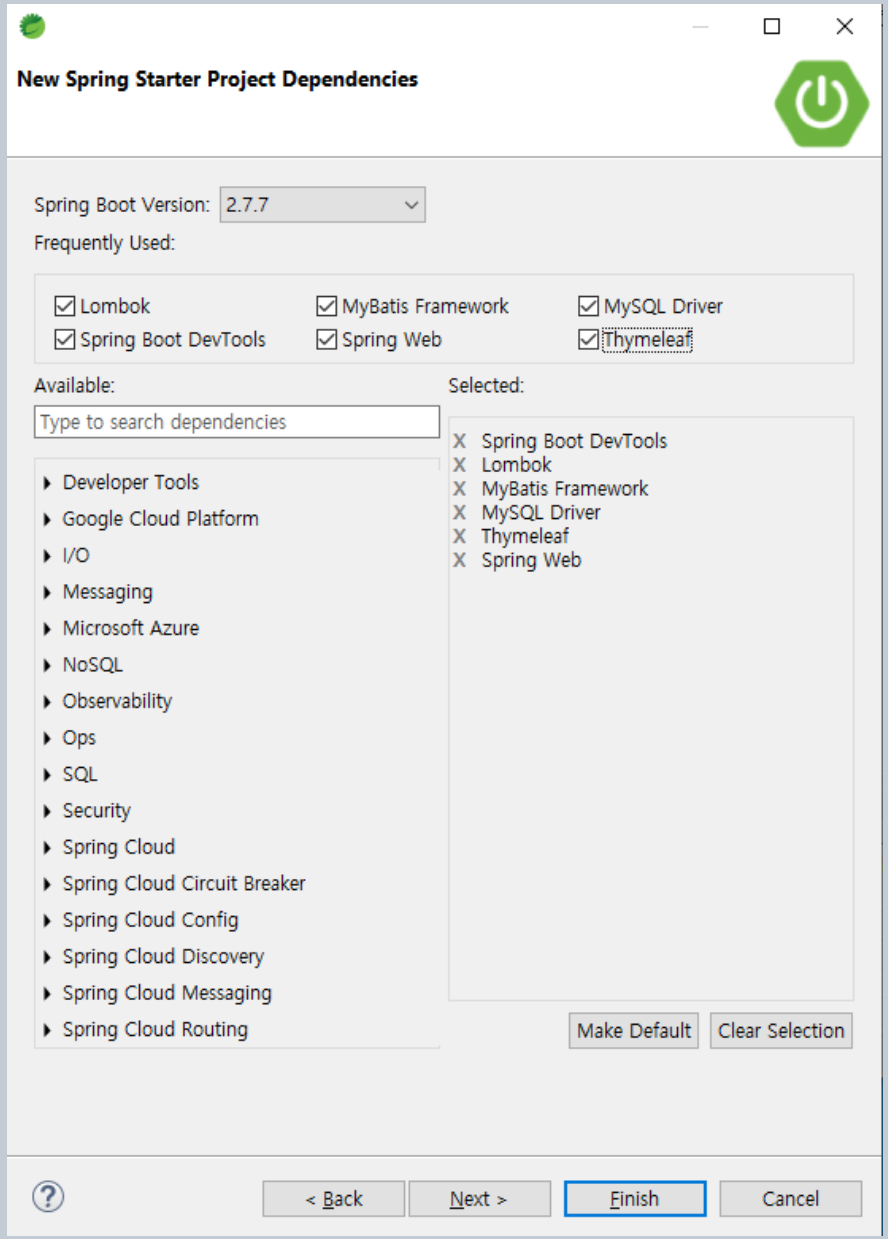
- 프로젝트 생성 및 구성
- 화면 구현
- 기능 구현
- 실행

과정 수행

1. 프로젝트 생성 및 구성

- 프로젝트 : BookStore
- WAS : Tomcat9
- DB : java2_bookstore
- Framework : SpringBoot, Mybatis
- Tools : Eclipse, Workbench





● **Application.properties**

```
server.servlet.context-path=/BookStore
server.port=8080
spring.thymeleaf.cache=false

#Mybatis 설정

#데이터베이스 정보
spring.datasource.url=jdbc:mysql://127.0.0.1:3306/java2_bookstore
spring.datasource.username=root
spring.datasource.password=1234
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

#MyBatis Mapper 경로설정
mybatis.mapper-locations=classpath:mappers/**/*.xml
```

● **BookStoreApplication**

```
package kr.co.bookstore;

import org.mybatis.spring.annotation.MapperScan;

@MapperScan("kr.co.bookstore.persistence")
@SpringBootApplication
public class BookStoreApplication {

    public static void main(String[] args) {
        SpringApplication.run(BookStoreApplication.class, args);
    }

}
```

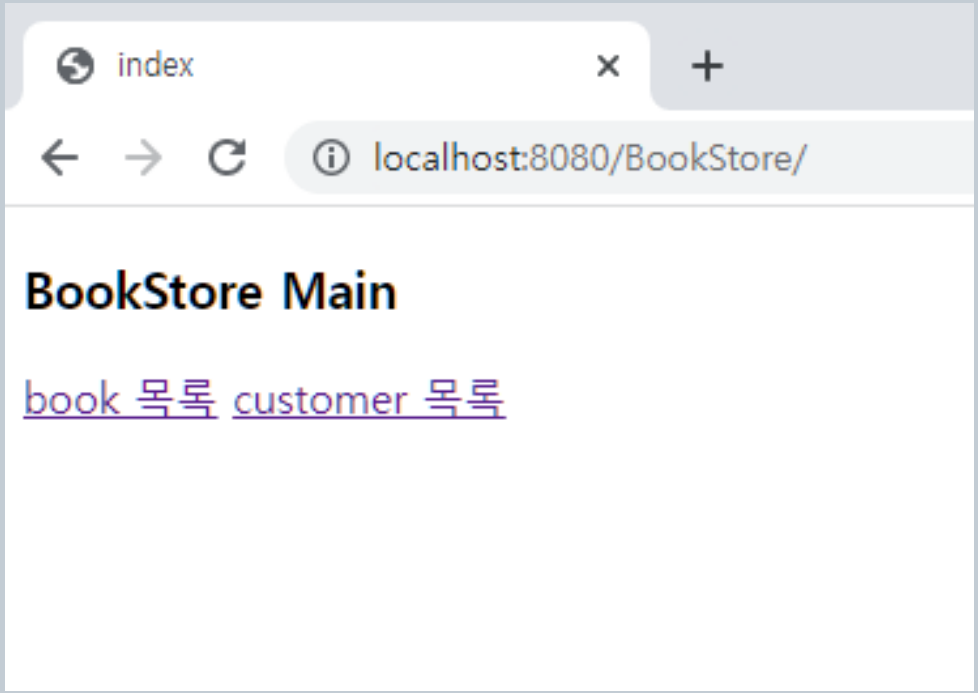
2. 화면 구현

- book(도서)

  목록, 등록, 수정 화면 구현

- customer(고객)

  목록, 등록, 수정 화면 구현




```html
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>index</title>
    </head>
<body>
    <h3>BookStore Main</h3>

    <a href="/BookStore/book/list">book 목록</a>
    <a href="/BookStore/customer/list">customer 목록</a>

</body>
</html>
```

● book, customer list 코드, 화면

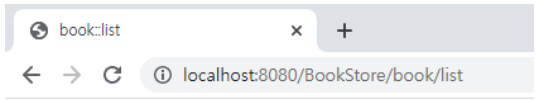
● book list

```html
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <meta charset="UTF-8">
        <title>book::list</title>
    </head>
<body>
    <h3>book 목록</h3>
    <a href="/BookStore">BookStore 메인</a>
    <a href="/BookStore/book/register">book 등록</a>

    <table border="1">
        <tr>
            <th>도서ID</th>
            <th>도서명</th>
            <th>출판사</th>
            <th>가격</th>
            <th>관리</th>
        </tr>
        <tr th:each="book:${books}">
            <td>[[${book.bookId}]]</td>
            <td>[[${book.bookName}]]</td>
            <td>[[${book.publisher}]]</td>
            <td>[[${book.price}]]</td>
            <td>
                <a th:href="@{/book/modify?(bookId=${book.bookId})}">수정</a>
                <a th:href="@{/book/delete?(bookId=${book.bookId})}">삭제</a>
            </td>
        </tr>
    </table>
</body>
</html>
```

● customer list

```html
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <meta charset="UTF-8">
        <title>customer::list</title>
    </head>
<body>
    <h3>customer 목록</h3>
    <a href="/BookStore">BookStore 메인</a>
    <a href="/BookStore/customer/register">customer 등록</a>

    <table border="1">
        <tr>
            <th>고객ID</th>
            <th>고객명</th>
            <th>주소</th>
            <th>휴대폰</th>
            <th>관리</th>
        </tr>
        <tr th:each="cust:${custs}">
            <td>[[${cust.custId}]]</td>
            <td>[[${cust.name}]]</td>
            <td>[[${cust.address}]]</td>
            <td>[[${cust.phone}]]</td>
            <td>
                <a th:href="@{/customer/modify?(custId=${cust.custId})}">수정</a>
                <a th:href="@{/customer/delete?(custId=${cust.custId})}">삭제</a>
            </td>
        </tr>
    </table>
</body>
</html>
```

● book list

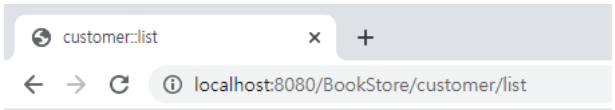


book 목록

BookStore 메인 book 등록

도서ID	도서명	출판사	가격	관리
1	축구의 역사	굿스포츠	7000	수정 삭제
2	축구아는 여자	나무수	13000	수정 삭제
3	축구의 이해	대한미디어	22000	수정 삭제
4	골프 바이블	대한미디어	35000	수정 삭제
5	피겨 교본	굿스포츠	8000	수정 삭제
6	역도 단계별기술	굿스포츠	6000	수정 삭제

● customer list



customer 목록

BookStore 메인 customer 등록

고객ID	고객명	주소	휴대폰	관리
1	박지성	영국 맨체스타	000-5000-0001	수정 삭제
2	김연아	대한민국 서울	000-6000-0001	수정 삭제
3	장미란	대한민국 강원도	000-7000-0001	수정 삭제
4	추신수	미국 클리블랜드	000-8000-0001	수정 삭제
5	이수근	서울특별시	010-1235-5221	수정 삭제

● book, customer register 코드

● book register

```html
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <meta charset="UTF-8">
        <title>book::register</title>
    </head>
<body>
    <h3>book 등록</h3>
    <a href="/BookStore">BookStore 메인</a>
    <a href="/BookStore/book/list">book 목록</a>

    <form action="/BookStore/book/register" method="post">
        <table border="1">
            <tr>
                <td>도서명</td>
                <td><input type="text" name="bookName"></td>
            </tr>
            <tr>
                <td>출판사</td>
                <td><input type="text" name="publisher"></td>
            </tr>
            <tr>
                <td>가격</td>
                <td><input type="number" name="price"></td>
            </tr>
            <tr>
                <td colspan="2" align="right"><input type="submit" value="등록"></td>
            </tr>
        </table>
    </form>
</body>
</html>
```

● customer register

```html
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <meta charset="UTF-8">
        <title>customer::register</title>
    </head>
<body>
    <h3>customer 등록</h3>
    <a href="/BookStore">BookStore 메인</a>
    <a href="/BookStore/customer/list">customer 목록</a>

    <form action="/BookStore/customer/register" method="post">
        <table border="1">
            <tr>
                <td>고객명</td>
                <td><input type="text" name="name"></td>
            </tr>
            <tr>
                <td>주소</td>
                <td><input type="text" name="address"></td>
            </tr>
            <tr>
                <td>휴대폰</td>
                <td><input type="text" name="phone"></td>
            </tr>
            <tr>
                <td colspan="2" align="right"><input type="submit" value="등록"></td>
            </tr>
        </table>
    </form>
</body>
</html>
```

- book, customer modify 코드

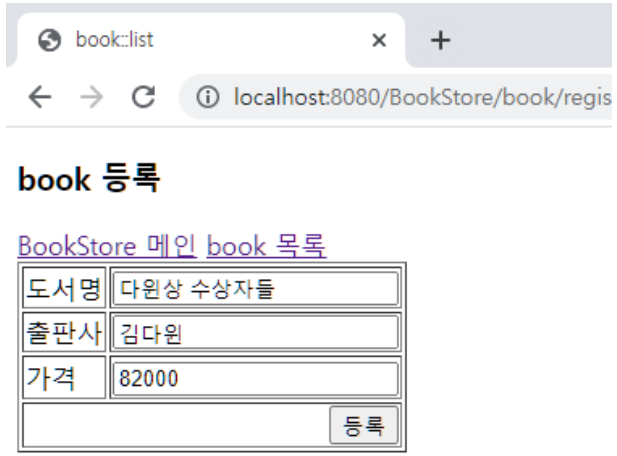
- book modify

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <meta charset="UTF-8">
        <title>book::modify</title>
    </head>
<body>
    <h3>book 수정</h3>
    <a href="/BookStore">BookStore 메인</a>
    <a href="/BookStore/book/list">book 목록</a>

    <form th:action="@{/book/modify?(bookId=${book.bookId})}" method="post">
        <table border="1">
            <tr>
                <td>도서ID</td>
                <td><input type="text" name="bookId" th:value="${book.bookId}" readonly="readonly"></td>
            </tr>
            <tr>
                <td>도서명</td>
                <td><input type="text" name="bookName" th:value="${book.bookName}"></td>
            </tr>
            <tr>
                <td>출판사</td>
                <td><input type="text" name="publisher" th:value="${book.publisher}"></td>
            </tr>
            <tr>
                <td>가격</td>
                <td><input type="number" name="price" th:value="${book.price}"></td>
            </tr>
            <tr>
                <td colspan="2" align="right"><input type="submit" value="등록"></td>
            </tr>
        </table>
    </form>
</body>
</html>
```

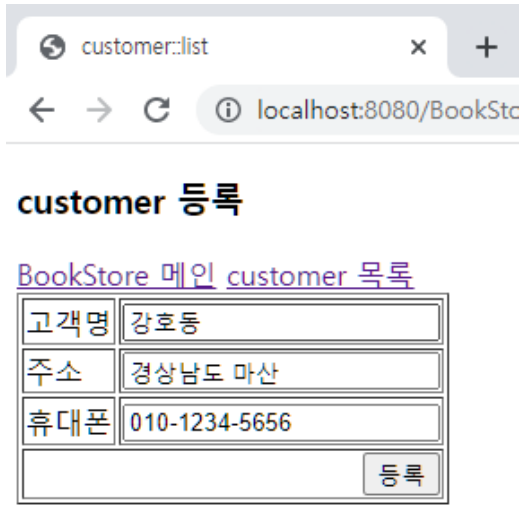
- customer modify

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
    <head>
        <meta charset="UTF-8">
        <title>customer::modify</title>
    </head>
<body>
    <h3>customer 수정</h3>
    <a href="/BookStore">BookStore 메인</a>
    <a href="/BookStore/customer/list">customer 목록</a>

    <form th:action="@{/customer/modify?(custId=${cust.custId})}" method="post">
        <table border="1">
            <tr>
                <td>고객ID</td>
                <td><input type="text" name="custId" th:value="${cust.custId}" readonly="readonly"></td>
            </tr>
            <tr>
                <td>고객명</td>
                <td><input type="text" name="name" th:value="${cust.name}"></td>
            </tr>
            <tr>
                <td>주소</td>
                <td><input type="text" name="address" th:value="${cust.address}"></td>
            </tr>
            <tr>
                <td>휴대폰</td>
                <td><input type="text" name="phone" th:value="${cust.phone}"></td>
            </tr>
            <tr>
                <td colspan="2" align="right"><input type="submit" value="등록"></td>
            </tr>
        </table>
    </form>
</body>
</html>
```

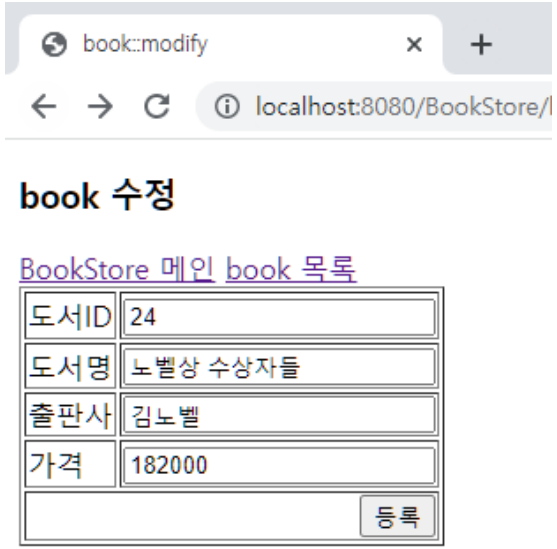
● book register



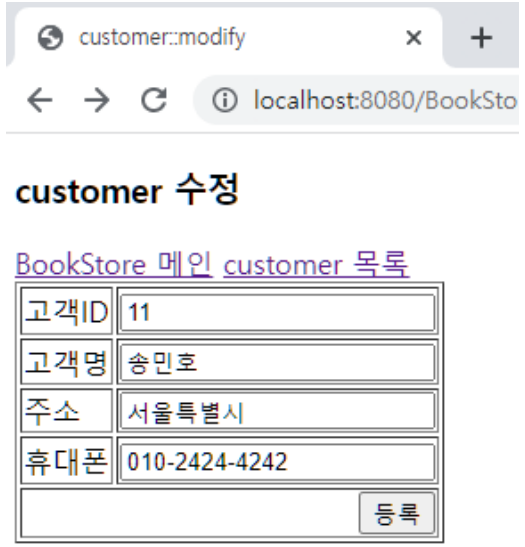
● customer register



● book modify



● customer modify



3. 기능 구현

- book(도서) 기본 CRUD
- customer(고객) 기본 CRUD
- Mybatis 사용

● book, customer mappers xml

● book

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

    <insert id="insertBook">
        INSERT INTO `book`(bookName, publisher, price) VALUES (#{bookName}, #{publisher}, #{price});
    </insert>

    <select id="selectBook" resultType="kr.co.bookstore.vo.BookVO">
    select * from `book` where `bookId` = #{bookId};
    </select>

    <select id="selectBooks" resultType="kr.co.bookstore.vo.BookVO">
        select * from `book`;
    </select>

    <update id="updateBook">
        update `book` set
                `bookName`=#{bookName},
                `publisher`=#{publisher},
                `price`=#{price}
        where
                `bookId`=#{bookId};
    </update>
    <delete id="deleteBook">
        delete from `book` where `bookId`=#{bookId};
    </delete>
</mapper>
```

● customer

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

    <insert id="insertCust">
        INSERT INTO `customer`(name, address, phone) VALUES (#{name}, #{address}, #{phone});
    </insert>

    <select id="selectCust" resultType="kr.co.bookstore.vo.CustVO">
    select * from `customer` where `custId` = #{custId};
    </select>

    <select id="selectCusts" resultType="kr.co.bookstore.vo.CustVO">
        select * from `customer`;
    </select>

    <update id="updateCust">
        update `customer` set
                `name`=#{name},
                `address`=#{address},
                `phone`=#{phone}
        where
                `custId`=#{custId};
    </update>
    <delete id="deleteCust">
        delete from `customer` where `custId`=#{custId};
    </delete>
</mapper>
```

- book

- VO

```java
package kr.co.bookstore.vo;

import lombok.AllArgsConstructor;

@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
@ToString
public class BookVO {
    private int bookId;
    private String bookName;
    private String publisher;
    private int price;
}
```

- DAO

```java
package kr.co.bookstore.dao;

import java.util.List;

@Repository
public class BookDAO {

    @Autowired
    private SqlSessionTemplate mybatis;

    public void insertBook(BookVO vo) {
        mybatis.insert("book.insertBook", vo);
    }

    public BookVO selectBook(int bookId) {
        return mybatis.selectOne("book.selectBook", bookId);
    }

    public List<BookVO> selectBooks() {
        return mybatis.selectList("book.selectBooks");
    }

    public void updateBook(BookVO vo) {
        mybatis.update("book.updateBook", vo);
    }

    public void deleteBook(int bookId) {
        mybatis.delete("book.deleteBook", bookId);
    }
}
```

- Service

```java
package kr.co.bookstore.service;

import java.util.List;

@Service
public class BookService {

    @Autowired
    private BookDAO dao;

    public void insertBook(BookVO vo) {
        dao.insertBook(vo);
    }

    public BookVO selectBook(int bookId) {
        return dao.selectBook(bookId);
    }

    public List<BookVO> selectBooks() {
        return dao.selectBooks();
    }

    public void updateBook(BookVO vo) {
        dao.updateBook(vo);
    }

    public void deleteBook(int bookId) {
        dao.deleteBook(bookId);
    }
}
```

- Controller

```java
@Controller
public class BookController {

    @Autowired
    private BookService service;

    @GetMapping("/book/list")
    public String list(Model model) {
        List<BookVO> books = service.selectBooks();
        model.addAttribute("books", books);
        return "/book/list";
    }

    @GetMapping("/book/register")
    public String register() {
        return "/book/register";
    }

    @PostMapping("/book/register")
    public String register(BookVO vo) {
        service.insertBook(vo);
        return "redirect:/book/list";
    }

    @GetMapping("/book/modify")
    public String modify(int bookId, Model model) {
        BookVO book = service.selectBook(bookId);
        model.addAttribute("book", book);
        return "/book/modify";
    }

    @PostMapping("/book/modify")
    public String modify(BookVO vo) {
        service.updateBook(vo);
        return "redirect:/book/list";
    }

    @GetMapping("/book/delete")
    public String delete(int bookId) {
        service.deleteBook(bookId);
        return "redirect:/book/list";
    }
```

- customer

- VO

```java
package kr.co.bookstore.vo;

import lombok.AllArgsConstructor;

@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
@ToString
public class CustVO {
    private int custId;
    private String name;
    private String address;
    private String phone;
}
```

- DAO

```java
package kr.co.bookstore.dao;

import java.util.List;

import org.mybatis.spring.SqlSessionTemplate;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;

import kr.co.bookstore.vo.CustVO;

@Repository
public class CustDAO {

    @Autowired
    private SqlSessionTemplate mybatis;

    public void insertCust(CustVO vo) {
        mybatis.insert("cust.insertCust", vo);
    }

    public CustVO selectCust(int custId) {
        return mybatis.selectOne("cust.selectCust", custId);
    }

    public List<CustVO> selectCusts() {
        return mybatis.selectList("cust.selectCusts");
    }

    public void updateCust(CustVO vo) {
        mybatis.update("cust.updateCust", vo);
    }

    public void deleteCust(int custId) {
        mybatis.delete("cust.deleteCust", custId);
    }
}
```

- Service

```java
package kr.co.bookstore.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import kr.co.bookstore.dao.CustDAO;
import kr.co.bookstore.vo.CustVO;

@Service
public class CustService {

    @Autowired
    private CustDAO dao;

    public void insertCust(CustVO vo) {
        dao.insertCust(vo);
    }

    public CustVO selectCust(int custId) {
        return dao.selectCust(custId);
    }

    public List<CustVO> selectCusts() {
        return dao.selectCusts();
    }

    public void updateCust(CustVO vo) {
        dao.updateCust(vo);
    }

    public void deleteCust(int custId) {
        dao.deleteCust(custId);
    }
}
```

- Controller

```java
@Controller
public class CustController {

    @Autowired
    private CustService service;

    @GetMapping("/customer/list")
    public String list(Model model) {
        List<CustVO> custs = service.selectCusts();
        model.addAttribute("custs", custs);
        return "/customer/list";
    }

    @GetMapping("/customer/register")
    public String register() {
        return "/customer/register";
    }

    @PostMapping("/customer/register")
    public String register(CustVO vo) {
        service.insertCust(vo);
        return "redirect:/customer/list";
    }

    @GetMapping("/customer/modify")
    public String modify(int custId, Model model) {
        CustVO cust = service.selectCust(custId);
        model.addAttribute("cust", cust);
        return "/customer/modify";
    }

    @PostMapping("/customer/modify")
    public String modify(CustVO vo) {
        service.updateCust(vo);
        return "redirect:/customer/list";
    }

    @GetMapping("/customer/delete")
    public String delete(int custId) {
        service.deleteCust(custId);
        return "redirect:/customer/list";
    }
```

4. 실행

· book 등록, 수정, 삭제 기능
동작 화면 첨부

· customer
등록, 수정, 삭제 기능
동작 화면 첨부

※ 목록, 등록, 수정 화면은 이전 단계에서 첨부

• book

• 등록

book::list

localhost:8080/BookStore/book/list

book 목록

BookStore 메인 book 등록

도서ID	도서명	출판사	가격	관리
1	축구의 역사	굿스포츠	7000	수정 삭제
2	축구아는 여자	나무수	13000	수정 삭제
3	축구의 이해	대한미디어	22000	수정 삭제
4	골프 바이블	대한미디어	35000	수정 삭제
5	피겨 교본	굿스포츠	8000	수정 삭제
6	역도 단계별기술	굿스포츠	6000	수정 삭제
24	다윈상 수상자들	김다윈	82000	수정 삭제
25	정보처리기사 필기+실기	공무원	42000	수정 삭제
26	나는 순하게 살기로 했다	진라면	19000	수정 삭제

HeidiSQL 12.1.0.6537

도움말

호스트: 127.0.0.1 데이터베이스: java2_boo

테이블: book 데이터

java2_bookstore.book: 9 다음 모두 보기 정렬 열 (4/4)

bookId	bookName	publisher	price
1	축구의 역사	굿스포츠	7,000
2	축구아는 여자	나무수	13,000
3	축구의 이해	대한미디어	22,000
4	골프 바이블	대한미디어	35,000
5	피겨 교본	굿스포츠	8,000
6	역도 단계별기술	굿스포츠	6,000
24	다윈상 수상자들	김다윈	82,000
25	정보처리기사 필기+실기	공무원	42,000
26	나는 순하게 살기로 했다	진라면	19,000

• 수정

book::list

localhost:8080/BookStore/book/list

book 목록

BookStore 메인 book 등록

도서ID	도서명	출판사	가격	관리
1	축구의 역사	굿스포츠	7000	수정 삭제
2	축구아는 여자	나무수	13000	수정 삭제
3	축구의 이해	대한미디어	22000	수정 삭제
4	골프 바이블	대한미디어	35000	수정 삭제
5	피겨 교본	굿스포츠	8000	수정 삭제
6	역도 단계별기술	굿스포츠	6000	수정 삭제
24	노벨상 수상자들	김노벨	182000	수정 삭제
25	정보처리기사 필기+실기	공무원	42000	수정 삭제
26	나는 순하게 살기로 했다	진라면	19000	수정 삭제

HeidiSQL 12.1.0.6537

도움말

호스트: 127.0.0.1 데이터베이스: java2_bookste

테이블: book 데이터

java2_bookstore.book: 9 다음 모두 보기 정렬 열 (4/4)

bookId	bookName	publisher	price
1	축구의 역사	굿스포츠	7,000
2	축구아는 여자	나무수	13,000
3	축구의 이해	대한미디어	22,000
4	골프 바이블	대한미디어	35,000
5	피겨 교본	굿스포츠	8,000
6	역도 단계별기술	굿스포츠	6,000
24	노벨상 수상자들	김노벨	182,000
25	정보처리기사 필기+실기	공무원	42,000
26	나는 순하게 살기로 했다	진라면	19,000

• 삭제

book::list

localhost:8080/BookStore/book/list

book 목록

BookStore 메인 book 등록

도서ID	도서명	출판사	가격	관리
1	축구의 역사	굿스포츠	7000	수정 삭제
2	축구아는 여자	나무수	13000	수정 삭제
3	축구의 이해	대한미디어	22000	수정 삭제
4	골프 바이블	대한미디어	35000	수정 삭제
5	피겨 교본	굿스포츠	8000	수정 삭제
6	역도 단계별기술	굿스포츠	6000	수정 삭제
25	정보처리기사 필기+실기	공무원	42000	수정 삭제
26	나는 순하게 살기로 했다	진라면	19000	수정 삭제

HeidiSQL 12.1.0.6537

도움말

호스트: 127.0.0.1 데이터베이스: java2_book

테이블: book 데이터

java2_bookstore.book: 8 다음 모두 보기 정렬 열 (4/4)

bookId	bookName	publisher	price
1	축구의 역사	굿스포츠	7,000
2	축구아는 여자	나무수	13,000
3	축구의 이해	대한미디어	22,000
4	골프 바이블	대한미디어	35,000
5	피겨 교본	굿스포츠	8,000
6	역도 단계별기술	굿스포츠	6,000
25	정보처리기사 필기+실기	공무원	42,000
26	나는 순하게 살기로 했다	진라면	19,000

• customer

• 등록

customer::list

localhost:8080/BookStore/customer/list

customer 목록

BookStore 메인 customer 등록

고객ID	고객명	주소	휴대폰	관리
1	박지성	영국 맨체스타	000-5000-0001	수정 삭제
2	김연아	대한민국 서울	000-6000-0001	수정 삭제
3	장미란	대한민국 강원도	000-7000-0001	수정 삭제
4	추신수	미국 클리블랜드	000-8000-0001	수정 삭제
5	이수근	서울특별시	010-1235-5221	수정 삭제
10	이승기	서울특별시	010-5252-8282	수정 삭제
11	강호동	경상남도 마산	010-2424-4242	수정 삭제
12	은지원	서울특별시	010-4646-7979	수정 삭제

- HeidiSQL 12.1.0.6537

도움말

호스트: 127.0.0.1 데이터베이스: java2_

테이블: customer 데이터

java2_bookstore.custome 다음 모두 보기 정렬 열 (

custId	name	address	phone
1	박지성	영국 맨체스타	000-5000-0001
2	김연아	대한민국 서울	000-6000-0001
3	장미란	대한민국 강원도	000-7000-0001
4	추신수	미국 클리블랜드	000-8000-0001
5	이수근	서울특별시	010-1235-5221
10	이승기	서울특별시	010-5252-8282
11	강호동	경상남도 마산	010-2424-4242
12	은지원	서울특별시	010-4646-7979

• 수정

customer::list

localhost:8080/BookStore/customer/list

customer 목록

BookStore 메인 customer 등록

고객ID	고객명	주소	휴대폰	관리
1	박지성	영국 맨체스타	000-5000-0001	수정 삭제
2	김연아	대한민국 서울	000-6000-0001	수정 삭제
3	장미란	대한민국 강원도	000-7000-0001	수정 삭제
4	추신수	미국 클리블랜드	000-8000-0001	수정 삭제
5	이수근	서울특별시	010-1235-5221	수정 삭제
10	이승기	서울특별시	010-5252-8282	수정 삭제
11	송민호	서울특별시	010-2424-4242	수정 삭제
12	은지원	서울특별시	010-4646-7979	수정 삭제

- HeidiSQL 12.1.0.6537

도움말

호스트: 127.0.0.1 데이터베이스: java2_boo

테이블: customer 데이터

java2_bookstore.custome 다음 모두 보기 정렬 열 (4/4)

custId	name	address	phone
1	박지성	영국 맨체스타	000-5000-0001
2	김연아	대한민국 서울	000-6000-0001
3	장미란	대한민국 강원도	000-7000-0001
4	추신수	미국 클리블랜드	000-8000-0001
5	이수근	서울특별시	010-1235-5221
10	이승기	서울특별시	010-5252-8282
11	송민호	서울특별시	010-2424-4242
12	은지원	서울특별시	010-4646-7979

• 삭제

customer::list

localhost:8080/BookStore/customer/list

customer 목록

BookStore 메인 customer 등록

고객ID	고객명	주소	휴대폰	관리
1	박지성	영국 맨체스타	000-5000-0001	수정 삭제
2	김연아	대한민국 서울	000-6000-0001	수정 삭제
3	장미란	대한민국 강원도	000-7000-0001	수정 삭제
4	추신수	미국 클리블랜드	000-8000-0001	수정 삭제
5	이수근	서울특별시	010-1235-5221	수정 삭제
10	이승기	서울특별시	010-5252-8282	수정 삭제
12	은지원	서울특별시	010-4646-7979	수정 삭제

- HeidiSQL 12.1.0.6537

도움말

호스트: 127.0.0.1 데이터베이스: java2_b

테이블: customer 데이터

java2_bookstore.custome 다음 모두 보기 정렬 열 (

custId	name	address	phone
1	박지성	영국 맨체스타	000-5000-0001
2	김연아	대한민국 서울	000-6000-0001
3	장미란	대한민국 강원도	000-7000-0001
4	추신수	미국 클리블랜드	000-8000-0001
5	이수근	서울특별시	010-1235-5221
10	이승기	서울특별시	010-5252-8282
12	은지원	서울특별시	010-4646-7979