모던 웹 애플리케이션 개발2

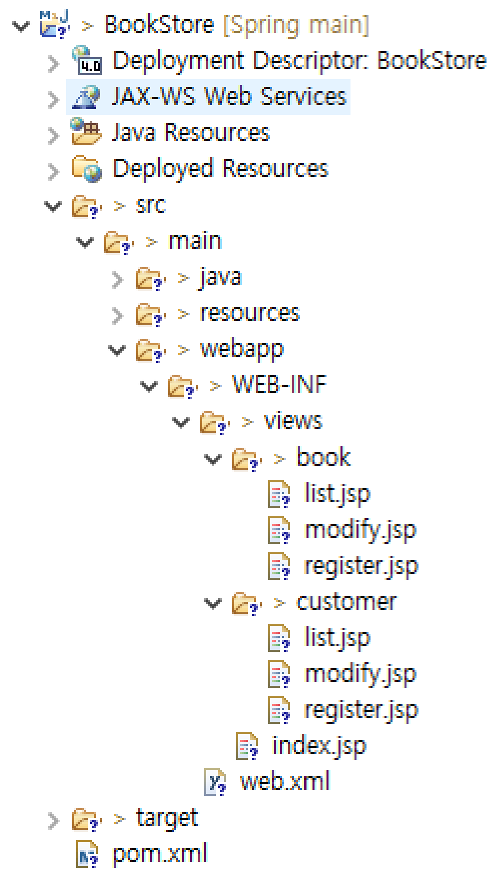
홍민준

2023-01-06 정기 수행평가

목차

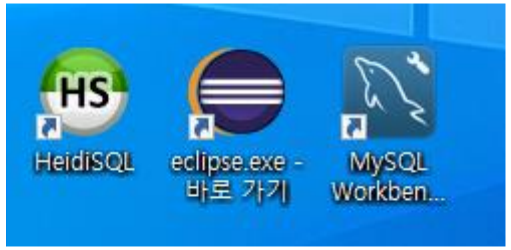
Q 1 – 프로젝트 생성 및 구성

- 프로젝트 구조



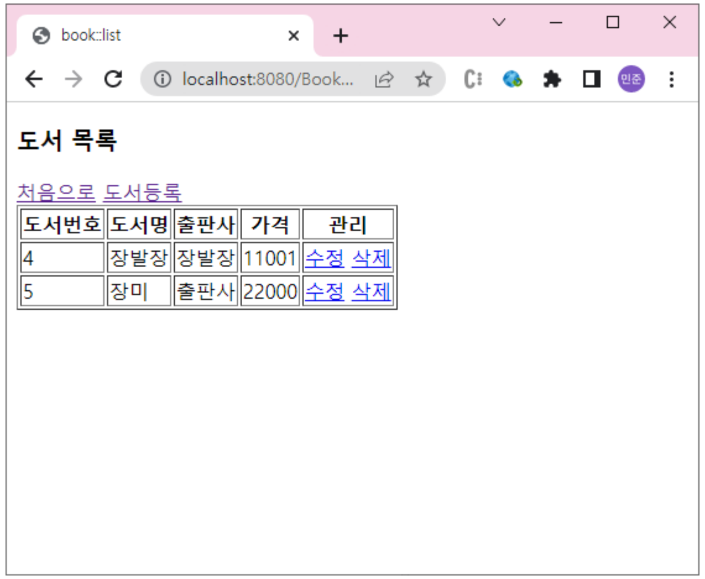
```
BookStore [Spring main]
  Deployment Descriptor: BookStore
  JAX-WS Web Services
  Java Resources
  Deployed Resources
  src
    main
      java
      resources
      webapp
        WEB-INF
          views
            book
              list.jsp
              modify.jsp
              register.jsp
            customer
              list.jsp
              modify.jsp
              register.jsp
          index.jsp
        web.xml
  target
  pom.xml
```

- 개발도구



Q2 – book 화면 구현

- book 목록

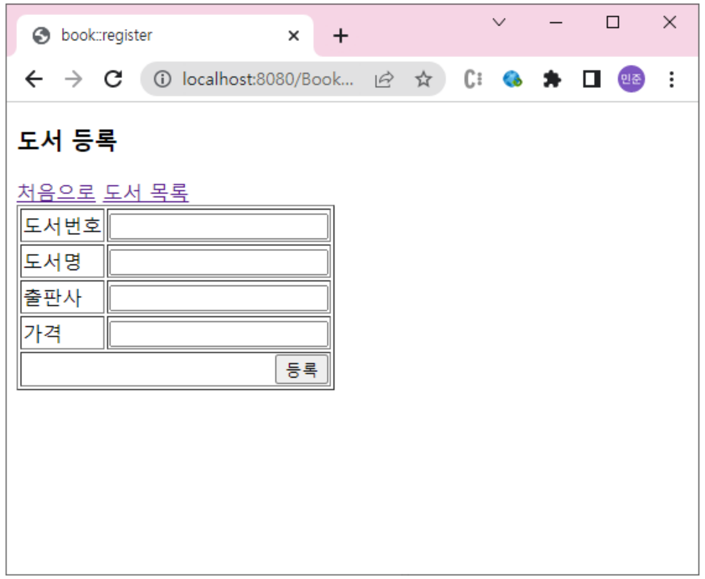


도서 목록

처음으로 도서등록

도서번호	도서명	출판사	가격	관리
4	장발장	장발장	11001	수정 삭제
5	장미	출판사	22000	수정 삭제

- book 등록

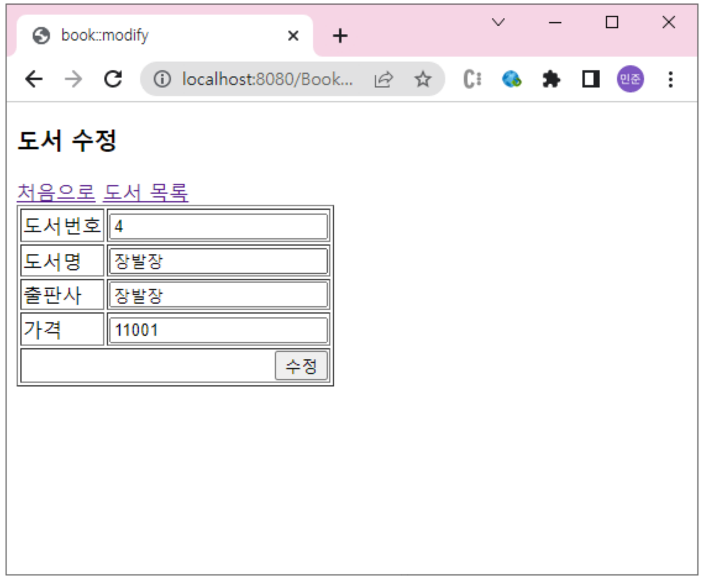


도서 등록

처음으로 도서 목록

도서번호	
도서명	
출판사	
가격	
	등록

- book 수정



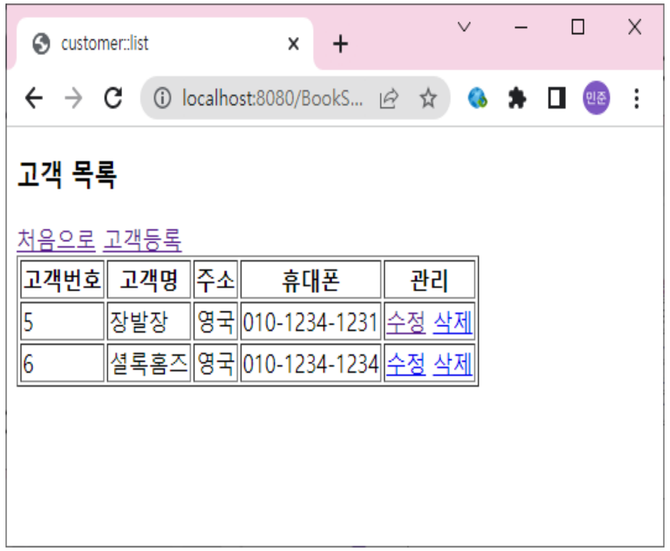
도서 수정

처음으로 도서 목록

도서번호	4
도서명	장발장
출판사	장발장
가격	11001
	수정

Q 2 – Customer 화면 구현

⊙ Customer 목록

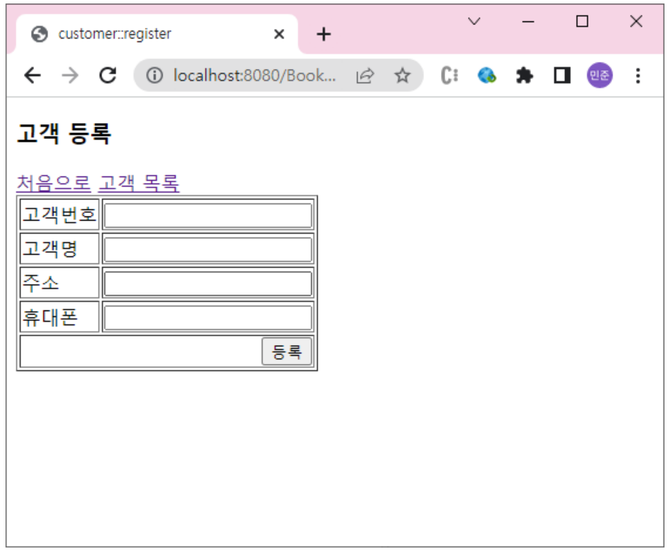


고객 목록

처음으로 고객등록

고객번호	고객명	주소	휴대폰	관리
5	장발장	영국	010-1234-1231	수정 삭제
6	셜록홈즈	영국	010-1234-1234	수정 삭제

⊙ Customer 등록

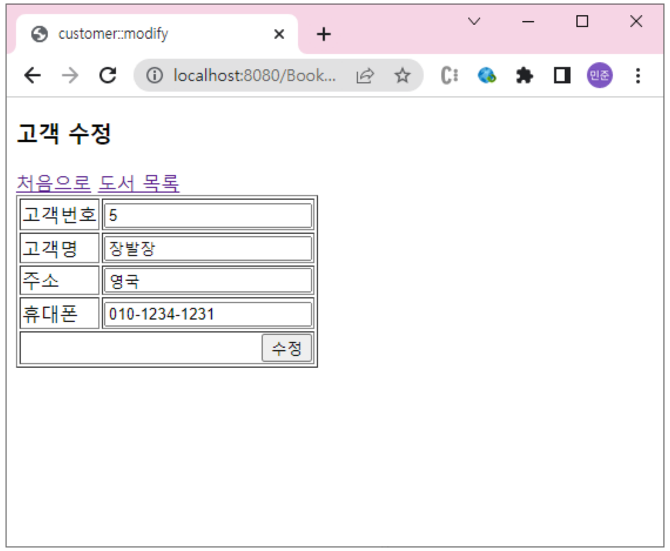


고객 등록

처음으로 고객 목록

고객번호	
고객명	
주소	
휴대폰	
	등록

⊙ Customer 수정



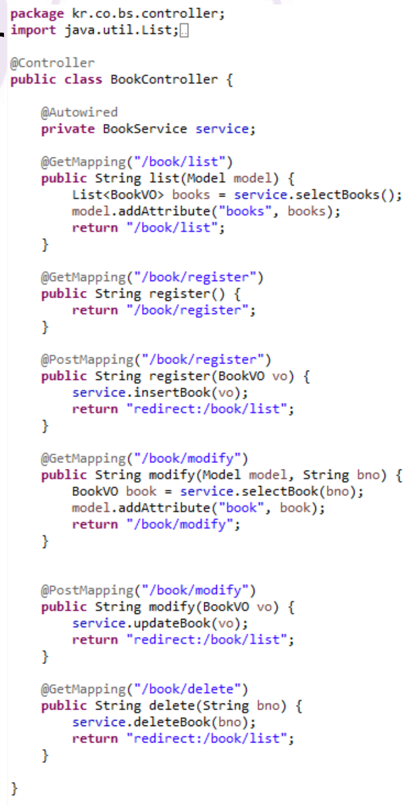
고객 수정

처음으로 도서 목록

고객번호	5
고객명	장발장
주소	영국
휴대폰	010-1234-1231
	수정

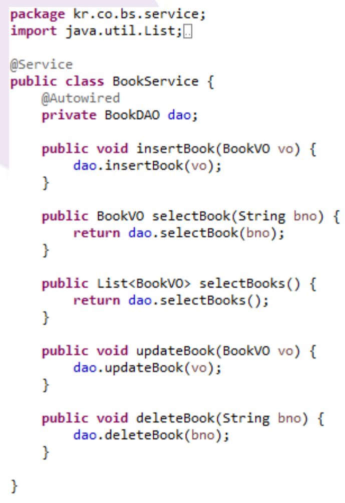
Q 3 – book 기능 구현

- book 코드:Controller



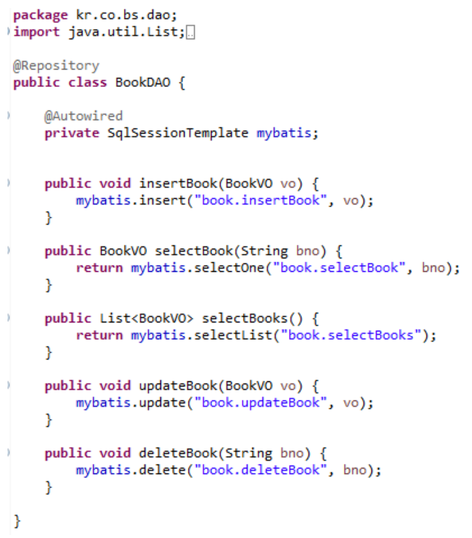
```
package kr.co.bs.controller;
import java.util.List;

@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(Model model, String bno) {
        BookVO book = service.selectBook(bno);
        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(String bno) {
        service.deleteBook(bno);
        return "redirect:/book/list";
    }

}
```

- book 코드:service



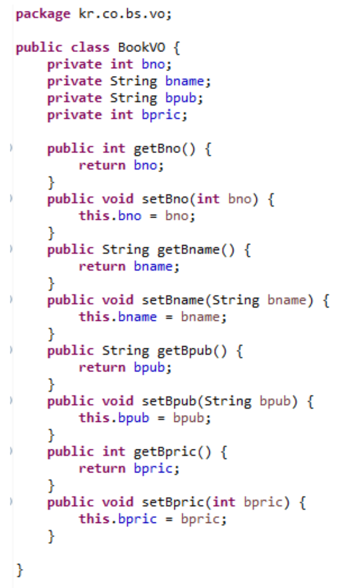
```
package kr.co.bs.service;
import java.util.List;

@Service
public class BookService {
    @Autowired
    private BookDAO dao;

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

    public BookVO selectBook(String bno) {
        return dao.selectBook(bno);
    }

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

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

    public void deleteBook(String bno) {
        dao.deleteBook(bno);
    }

}
```

- book 코드:dao



```
package kr.co.bs.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(String bno) {
        return mybatis.selectOne("book.selectBook", bno);
    }

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

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

    public void deleteBook(String bno) {
        mybatis.delete("book.deleteBook", bno);
    }

}
```

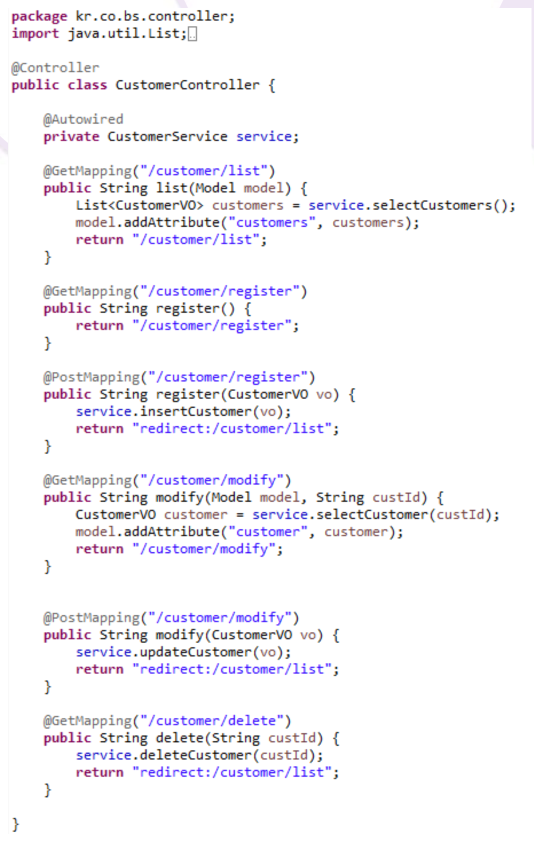
- book 코드:VO



```
package kr.co.bs.vo;

public class BookVO {
    private int bno;
    private String bname;
    private String bpub;
    private int bpric;

    public int getBno() {
        return bno;
    }
    public void setBno(int bno) {
        this.bno = bno;
    }
    public String getBname() {
        return bname;
    }
    public void setBname(String bname) {
        this.bname = bname;
    }
    public String getBpub() {
        return bpub;
    }
    public void setBpub(String bpub) {
        this.bpub = bpub;
    }
    public int getBpric() {
        return bpric;
    }
    public void setBpric(int bpric) {
        this.bpric = bpric;
    }

}
```

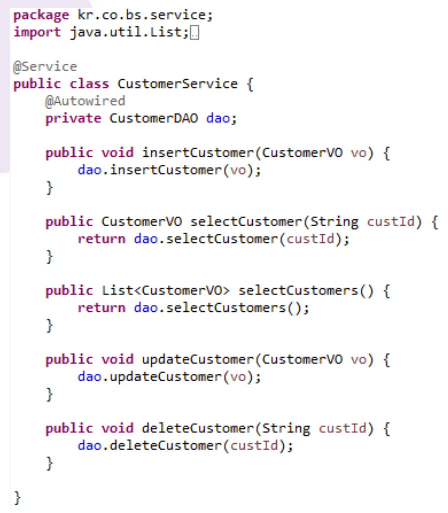
Q 3 – customer 기능 구현

- Customer코드:Controller



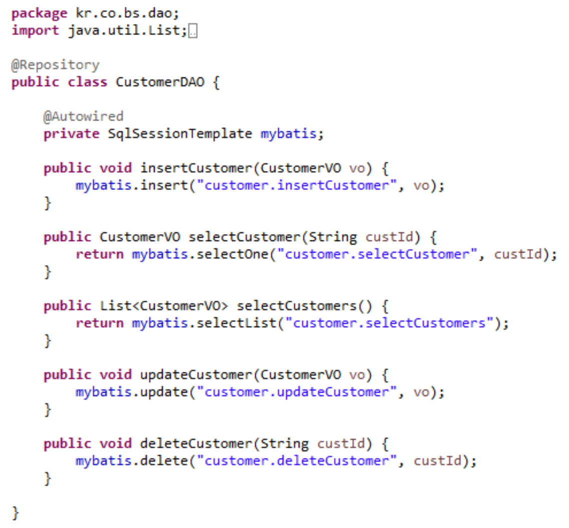
```
package kr.co.bs.controller;
import java.util.List;

@Controller
public class CustomerController {

    @Autowired
    private CustomerService service;

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

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

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

    @GetMapping("/customer/modify")
    public String modify(Model model, String custId) {
        CustomerVO customer = service.selectCustomer(custId);
        model.addAttribute("customer", customer);
        return "/customer/modify";
    }


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

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

}
```

- Customer코드:sercvice



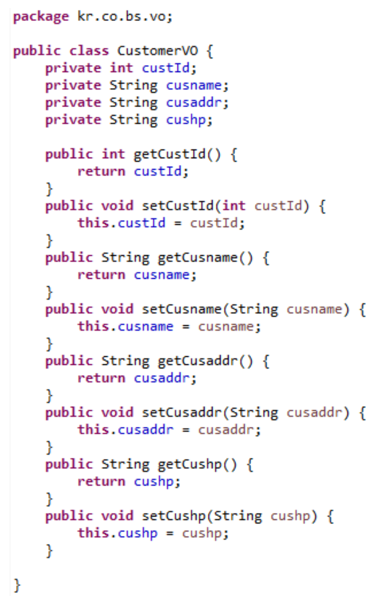
```
package kr.co.bs.service;
import java.util.List;

@Service
public class CustomerService {
    @Autowired
    private CustomerDAO dao;

    public void insertCustomer(CustomerVO vo) {
        dao.insertCustomer(vo);
    }

    public CustomerVO selectCustomer(String custId) {
        return dao.selectCustomer(custId);
    }

    public List<CustomerVO> selectCustomers() {
        return dao.selectCustomers();
    }

    public void updateCustomer(CustomerVO vo) {
        dao.updateCustomer(vo);
    }

    public void deleteCustomer(String custId) {
        dao.deleteCustomer(custId);
    }

}
```

- Customer코드:dao



```
package kr.co.bs.dao;
import java.util.List;

@Repository
public class CustomerDAO {

    @Autowired
    private SqlSessionTemplate mybatis;

    public void insertCustomer(CustomerVO vo) {
        mybatis.insert("customer.insertCustomer", vo);
    }

    public CustomerVO selectCustomer(String custId) {
        return mybatis.selectOne("customer.selectCustomer", custId);
    }

    public List<CustomerVO> selectCustomers() {
        return mybatis.selectList("customer.selectCustomers");
    }

    public void updateCustomer(CustomerVO vo) {
        mybatis.update("customer.updateCustomer", vo);
    }

    public void deleteCustomer(String custId) {
        mybatis.delete("customer.deleteCustomer", custId);
    }

}
```

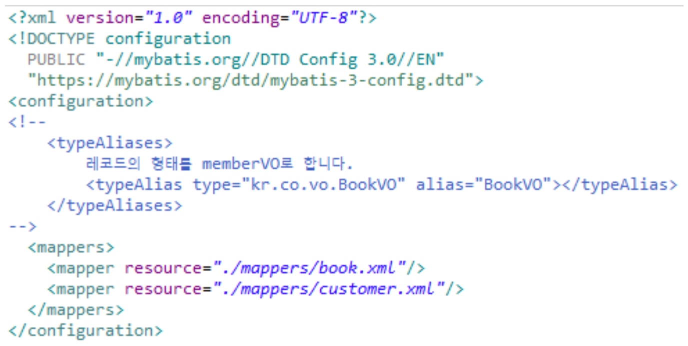
- Customer코드:VO



```
package kr.co.bs.vo;

public class CustomerVO {
    private int custId;
    private String cusname;
    private String cusaddr;
    private String cushp;

    public int getCustId() {
        return custId;
    }
    public void setCustId(int custId) {
        this.custId = custId;
    }
    public String getCusname() {
        return cusname;
    }
    public void setCusname(String cusname) {
        this.cusname = cusname;
    }
    public String getCusaddr() {
        return cusaddr;
    }
    public void setCusaddr(String cusaddr) {
        this.cusaddr = cusaddr;
    }
    public String getCushp() {
        return cushp;
    }
    public void setCushp(String cushp) {
        this.cushp = cushp;
    }

}
```

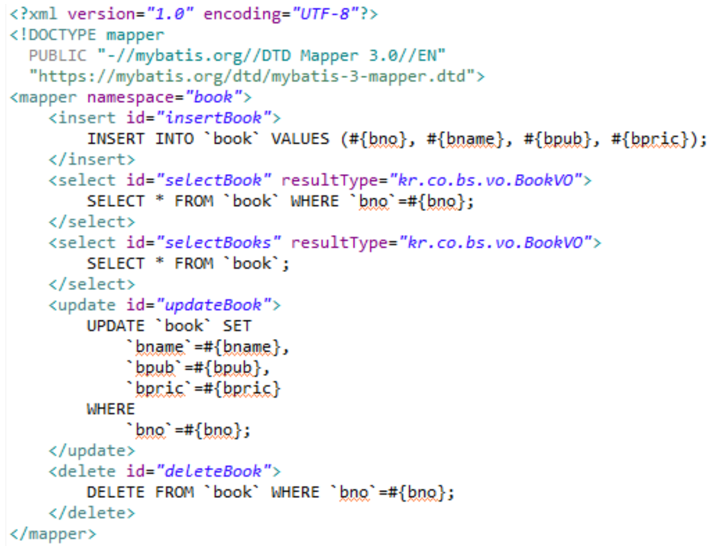
Q 3 – 기능 구현 Mybatis

Context



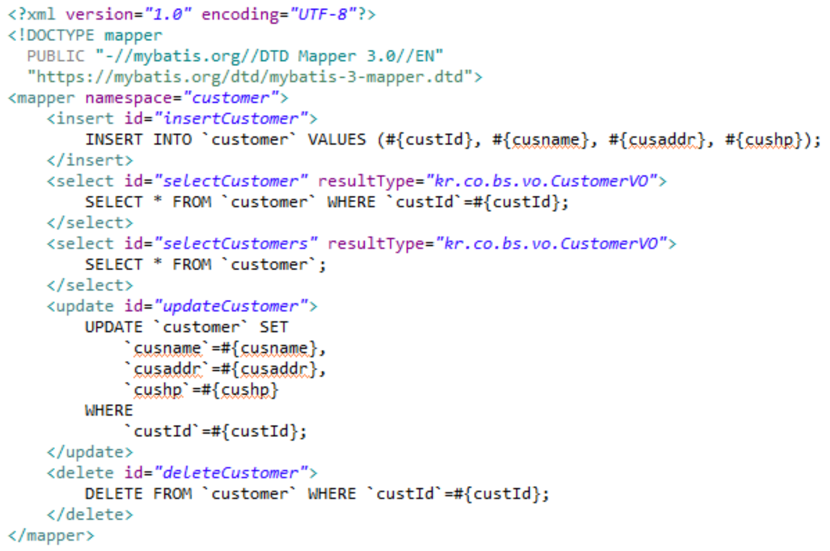
```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration
  PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
  "https://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
<!--
    <typeAliases>
        레코드의 형태를 memberVO로 합니다.
        <typeAlias type="kr.co.vo.BookVO" alias="BookVO"></typeAlias>
    </typeAliases>
-->
  <mappers>
    <mapper resource="./mappers/book.xml"/>
    <mapper resource="./mappers/customer.xml"/>
  </mappers>
</configuration>
```

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` VALUES (#{bno}, #{bname}, #{bpub}, #{bpric});
    </insert>
    <select id="selectBook" resultType="kr.co.bs.vo.BookVO">
        SELECT * FROM `book` WHERE `bno`=#{bno};
    </select>
    <select id="selectBooks" resultType="kr.co.bs.vo.BookVO">
        SELECT * FROM `book`;
    </select>
    <update id="updateBook">
        UPDATE `book` SET
            `bname`=#{bname},
            `bpub`=#{bpub},
            `bpric`=#{bpric}
        WHERE
            `bno`=#{bno};
    </update>
    <delete id="deleteBook">
        DELETE FROM `book` WHERE `bno`=#{bno};
    </delete>
</mapper>
```

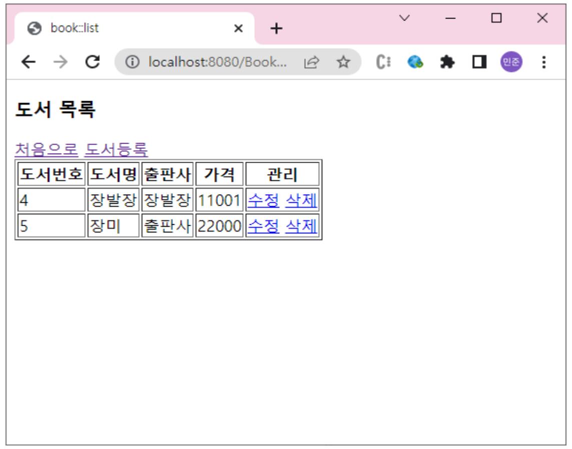
Customer



```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="customer">
    <insert id="insertCustomer">
        INSERT INTO `customer` VALUES (#{custId}, #{cusname}, #{cusaddr}, #{cushp});
    </insert>
    <select id="selectCustomer" resultType="kr.co.bs.vo.CustomerVO">
        SELECT * FROM `customer` WHERE `custId`=#{custId};
    </select>
    <select id="selectCustomers" resultType="kr.co.bs.vo.CustomerVO">
        SELECT * FROM `customer`;
    </select>
    <update id="updateCustomer">
        UPDATE `customer` SET
            `cusname`=#{cusname},
            `cusaddr`=#{cusaddr},
            `cushp`=#{cushp}
        WHERE
            `custId`=#{custId};
    </update>
    <delete id="deleteCustomer">
        DELETE FROM `customer` WHERE `custId`=#{custId};
    </delete>
</mapper>
```

Q 4 – book 실행

book 목록 동작



book::list

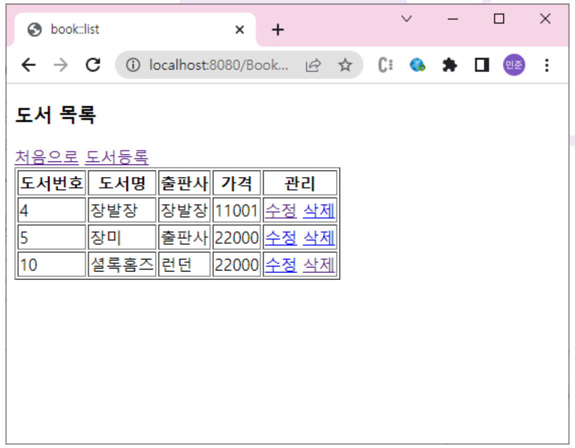
localhost:8080/Book...

도서 목록

처음으로 도서등록

도서번호	도서명	출판사	가격	관리
4	장발장	장발장	11001	수정 삭제
5	장미	출판사	22000	수정 삭제

book 등록 동작



book::list

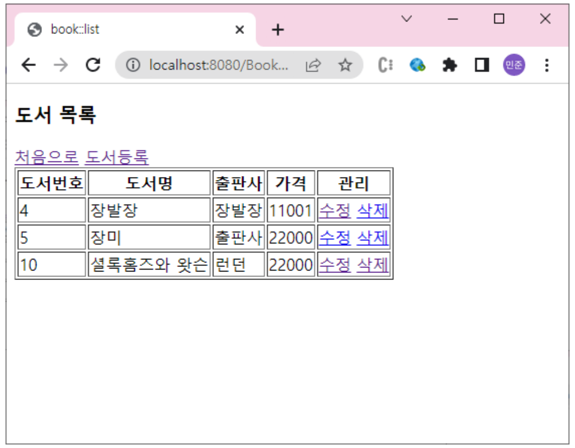
localhost:8080/Book...

도서 목록

처음으로 도서등록

도서번호	도서명	출판사	가격	관리
4	장발장	장발장	11001	수정 삭제
5	장미	출판사	22000	수정 삭제
10	셜록홈즈	런던	22000	수정 삭제

book 수정 동작



book::list

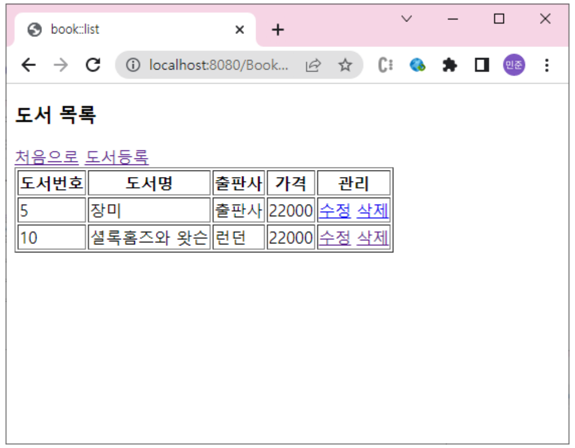
localhost:8080/Book...

도서 목록

처음으로 도서등록

도서번호	도서명	출판사	가격	관리
4	장발장	장발장	11001	수정 삭제
5	장미	출판사	22000	수정 삭제
10	셜록홈즈와 왓슨	런던	22000	수정 삭제

book 삭제 동작



book::list

localhost:8080/Book...

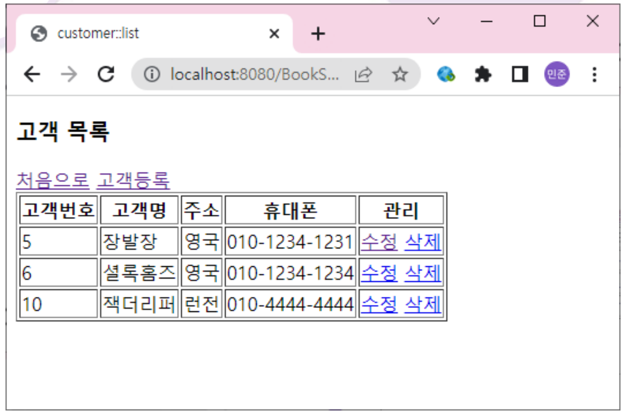
도서 목록

처음으로 도서등록

도서번호	도서명	출판사	가격	관리
5	장미	출판사	22000	수정 삭제
10	셜록홈즈와 왓슨	런던	22000	수정 삭제

Q 4 – customer 실행

- customer 등록 동작



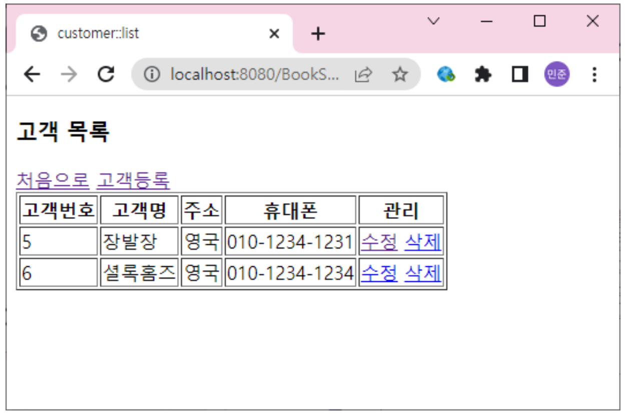
customer::list
localhost:8080/BookS...

고객 목록

처음으로 고객등록

고객번호	고객명	주소	휴대폰	관리
5	장발장	영국	010-1234-1231	수정 삭제
6	셜록홈즈	영국	010-1234-1234	수정 삭제
10	잭더리퍼	런전	010-4444-4444	수정 삭제

- customer 목록 동작



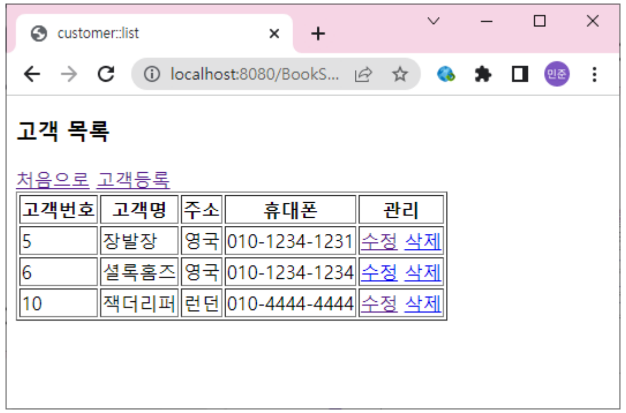
customer::list
localhost:8080/BookS...

고객 목록

처음으로 고객등록

고객번호	고객명	주소	휴대폰	관리
5	장발장	영국	010-1234-1231	수정 삭제
6	셜록홈즈	영국	010-1234-1234	수정 삭제

- customer 수정 동작



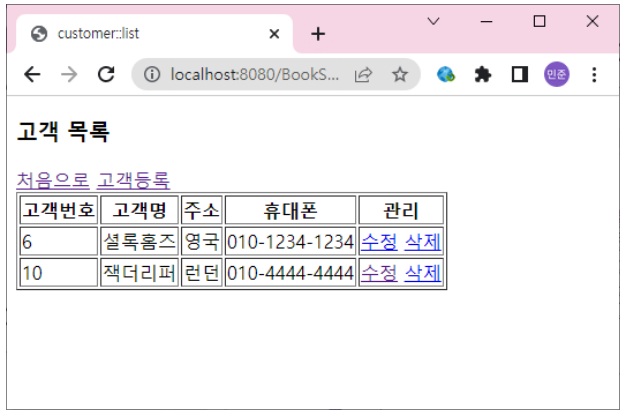
customer::list
localhost:8080/BookS...

고객 목록

처음으로 고객등록

고객번호	고객명	주소	휴대폰	관리
5	장발장	영국	010-1234-1231	수정 삭제
6	셜록홈즈	영국	010-1234-1234	수정 삭제
10	잭더리퍼	런던	010-4444-4444	수정 삭제

- customer 삭제 동작



customer::list
localhost:8080/BookS...

고객 목록

처음으로 고객등록

고객번호	고객명	주소	휴대폰	관리
6	셜록홈즈	영국	010-1234-1234	수정 삭제
10	잭더리퍼	런던	010-4444-4444	수정 삭제