




모던 웹 애플리케이션 개발2

2023/01/04(수)

김지홍

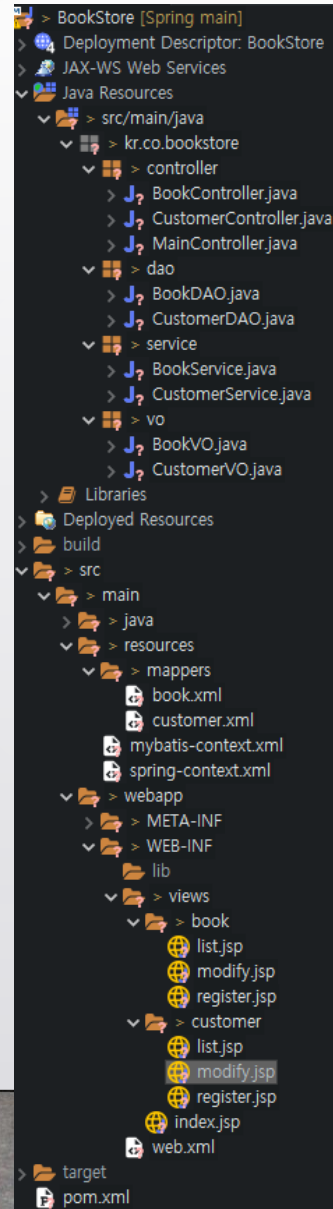


목차

1. 프로젝트 생성 및 구성
2. 화면 구현
3. 기능 구현
4. 실행

1. 프로젝트 생성 및 구성

3



2. 화면 구현

4

BookStore

[도서목록](#) [고객목록](#)

도서목록

[처음으로 도서등록](#)

도서번호	도서명	출판사	가격	관리
1	축구의 역사2	굿스포츠2	70000	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
7	야구의 추억	이상미디어	20000	수정 삭제
8	야구를 부탁해	이상미디어	13000	수정 삭제
9	올림픽 이야기	삼성당	7500	수정 삭제
10	Olympic Champions	Pearson	13000	수정 삭제
103	HTML/CSS	웹프로그래밍	300002	수정 삭제

고객목록

[처음으로 고객등록](#)

고객번호	고객명	주소	휴대폰	관리
1	박지성	영국 맨체스터	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	박세리	대한민국 부산		수정 삭제
6	이단계사	광주	010-1234	수정 삭제
13	손흥민	영국	011-0000-0002	수정 삭제

도서등록

[처음으로 도서목록](#)

도서명	<input type="text"/>
출판사	<input type="text"/>
가격	<input type="text"/>
<input type="button" value="등록"/>	

도서수정

[처음으로 도서목록](#)

도서번호	<input type="text" value="1"/>
도서명	<input type="text" value="축구의 역사2"/>
출판사	<input type="text" value="굿스포츠2"/>
가격	<input type="text" value="70000"/>
<input type="button" value="수정"/>	

고객등록

[처음으로 고객목록](#)

고객명	<input type="text"/>
주소	<input type="text"/>
휴대폰	<input type="text"/>
<input type="button" value="등록"/>	

고객수정

[처음으로 고객목록](#)

고객번호	<input type="text" value="1"/>
고객명	<input type="text" value="박지성"/>
주소	<input type="text" value="영국 맨체스터"/>
휴대폰	<input type="text" value="000-5000-0001"/>
<input type="button" value="수정"/>	

3. 기능 구현

5

pom.xml

```
<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis -->
<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis</artifactId>
  <version>3.5.11</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis-spring -->
<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis-spring</artifactId>
  <version>3.0.1</version>
</dependency>
```

spring-context.xml

```
<!-- 데이터베이스(커넥션풀) 설정 -->
<bean id="dataSource" class="org.apache.commons.dbcp2.BasicDataSource" destroy-method="close">
  <property name="driverClassName" value="com.mysql.cj.jdbc.Driver" />
  <property name="url" value="jdbc:mysql://localhost:3306/java1_bookstore?characterEncoding=UTF-8" />
  <property name="username" value="root" />
  <property name="password" value="1234" />

  <property name="initialSize" value="0"/>
  <property name="minIdle" value="0"/>
  <property name="maxIdle" value="10"/>
  <property name="maxTotal" value="10"/>
  <property name="maxWaitMillis" value="1000"/>
</bean>
<!-- Mybatis 설정 -->
<bean id="sqlSessionFactoryBean" class="org.mybatis.spring.SqlSessionFactoryBean">
  <property name="dataSource" ref="dataSource"/>
  <property name="configLocation" value="classpath:mybatis-context.xml"/>
</bean>
<!-- Mybatis-Spring 설정 -->
<bean id="sqlSessionTemplate" class="org.mybatis.spring.SqlSessionTemplate">
  <constructor-arg ref="sqlSessionFactoryBean"/>
</bean>
```

pom.xml에 mybatis 를 추가 spring-context.xml 에 설정을 해준다

3. 기능 구현

6

src > main > resources > 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>
  <mappers>
    <mapper resource="./mappers/book.xml"/>
    <mapper resource="./mappers/customer.xml"/>
  </mappers>
</configuration>
```

src > main > resources > mappers > 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">
  <select id="selectBooks" resultType="kr.co.bookstore.vo.BookVO">
    SELECT * FROM `book`;
  </select>
  <select id="selectBook" resultType="kr.co.bookstore.vo.BookVO" >
    SELECT * FROM `book` WHERE `bookId` = #{bookId};
  </select>
  <insert id="insertBook">
    INSERT INTO `book` SET `bookName` = #{bookName}, `publisher` = #{publisher}, `price` = #{price}
  </insert>
  <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>
```

BookVo

```
public class BookVO {
    private int bookId;
    private String bookName;
    private String publisher;
    private int price;

    public int getBookId() {
        return bookId;
    }
    public void setBookId(int bookId) {
        this.bookId = bookId;
    }
    public String getBookName() {
        return bookName;
    }
    public void setBookName(String bookName) {
        this.bookName = bookName;
    }
    public String getPublisher() {
        return publisher;
    }
    public void setPublisher(String publisher) {
        this.publisher = publisher;
    }
    public int getPrice() {
        return price;
    }
    public void setPrice(int price) {
        this.price = price;
    }
}
```

mybatis-context.xml 에 sql 설정을 할 파일을 mapper에 등록하고 각 파일에 sql을 사용한다.

3. 기능 구현

7

src > main > resources > mappers > 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">
  <select id="selectCustomers" resultType="kr.co.bookstore.vo.CustomerVO">
    SELECT * FROM `customer`;
  </select>
  <select id="selectCustomer" resultType="kr.co.bookstore.vo.CustomerVO" >
    SELECT * FROM `customer` WHERE `custId` = #{custId};
  </select>
  <insert id="insertCustomer">
    INSERT INTO `customer` SET `name` = #{name}, `address` = #{address}, `phone` = #{phone};
  </insert>
  <update id="updateCustomer">
    UPDATE `customer` SET `name` = #{name}, `address` = #{address}, `phone` = #{phone} WHERE `custId` = #{custId};
  </update>
  <delete id="deleteCustomer">
    DELETE FROM `customer` WHERE `custId` = #{custId};
  </delete>
</mapper>
```

CustomerVo

```
public class CustomerVO {
    private int custId;
    private String name;
    private String address;
    private String phone;

    public int getCustId() {
        return custId;
    }
    public void setCustId(int custId) {
        this.custId = custId;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getAddress() {
        return address;
    }
    public void setAddress(String address) {
        this.address = address;
    }
    public String getPhone() {
        return phone;
    }
    public void setPhone(String phone) {
        this.phone = phone;
    }
}
```

BookDAO

```
@Repository
public class BookDAO {

    @Autowired
    private SqlSessionTemplate mb;

    public List<BookVO> selectBooks(){
        return mb.selectList("book.selectBooks");
    }
    public BookVO selectBook(String bookId) {
        return mb.selectOne("book.selectBook", bookId);
    }
    public void insertBook(BookVO vo) {
        mb.insert("book.insertBook", vo);
    }
    public void updateBook(BookVO vo) {
        mb.insert("book.updateBook", vo);
    }
    public void deleteBook(String bookId) {
        mb.delete("book.deleteBook", bookId);
    }
}
```

CustomerDAO

```
@Repository
public class CustomerDAO {

    @Autowired
    private SqlSessionTemplate mb;

    public List<CustomerVO> selectCustomers() {
        return mb.selectList("customer.selectCustomers");
    }
    public CustomerVO selectCustomer(String custId) {
        return mb.selectOne("customer.selectCustomer", custId);
    }
    public void insertCustomer(CustomerVO vo) {
        mb.insert("customer.insertCustomer", vo);
    }
    public void updateCustomer(CustomerVO vo) {
        mb.update("customer.updateCustomer", vo);
    }
    public void deleteCustoemr(String custId) {
        mb.delete("customer.deleteCustomer", custId);
    }
}
```

Sqlsession을 불러와 앞선 mapper에 등록한 namespace.key 값으로 sql을 호출

BookService

```
@Service
public class BookService {

    @Autowired
    private BookDAO dao;

    public List<BookVO> selectBooks(){
        return dao.selectBooks();
    }
    public BookVO selectBook(String bookId) {
        return dao.selectBook(bookId);
    }
    public void insertBook(BookVO vo) {
        dao.insertBook(vo);
    }

    public void updateBook(BookVO vo) {
        dao.updateBook(vo);
    }
    public void deleteBook(String bookId) {
        dao.deleteBook(bookId);
    }
}
```

CustomerService

```
@Service
public class CustomerService {

    @Autowired
    private CustomerDAO dao;

    public List<CustomerVO> selectCustomers() {
        return dao.selectCustomers();
    }
    public CustomerVO selectCustomer(String custId) {
        return dao.selectCustomer(custId);
    }
    public void insertCustomer(CustomerVO vo) {
        dao.insertCustomer(vo);
    }
    public void updateCustomer(CustomerVO vo) {
        dao.updateCustomer(vo);
    }
    public void deleteCustoemr(String custId) {
        dao.deleteCustoemr(custId);
    }
}
```

3. 기능 구현

10

MainController

```
@Controller
public class MainController {

    @GetMapping(value= {"/", "/index"})
    public String index() {
        return "/index";
    }
}
```

CustomerController

```
@Controller
public class CustomerController {
    @Autowired
    CustomerService service;

    @GetMapping("/customer/list")
    public String CustomerList(Model m) {
        m.addAttribute("customers", service.selectCustomers());
        return "/customer/list";
    }

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

    @GetMapping("/customer/modify")
    public String CustomerModify(String custId, Model m) {
        m.addAttribute("customer", service.selectCustomer(custId));
        return "/customer/modify";
    }
    @PostMapping("/customer/modify")
    public String CustomerModify(CustomerVO vo) {
        service.updateCustomer(vo);
        return "redirect:/customer/list";
    }

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

BookController

```
@Controller
public class BookController {
    @Autowired
    BookService service;

    @GetMapping("/book/list")
    public String bookList(Model m) {
        m.addAttribute("books", service.selectBooks());
        return "/book/list";
    }

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

    @GetMapping("/book/modify")
    public String bookModify(String bookId, Model m) {
        m.addAttribute("book", service.selectBook(bookId));
        return "/book/modify";
    }
    @PostMapping("/book/modify")
    public String bookModify(BookVO vo) {
        service.updateBook(vo);
        return "redirect:/book/list";
    }

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

4. 실행

11

도서 등록

도서등록

[처음으로 도서목록](#)

도서명	테스트도서
출판사	테스트출판사
가격	75000
<div>등록</div>	

도서 수정

도서수정

[처음으로 도서목록](#)

도서번호	105
도서명	테스트도서2
출판사	테스트출판사2
가격	80000
<div>수정</div>	

도서 삭제

105	테스트도서2	테스트출판사2	80000	수정 삭제
-----	--------	---------	-------	---------------------------------------

10	Olympic Champions	Pearson	13000	수정 삭제
103	HTML/CSS	웹프로그래밍	300002	수정 삭제

도서목록

[처음으로 도서등록](#)

도서번호	도서명	출판사	가격	관리
1	축구의 역사2	굿스포츠2	70000	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
7	야구의 추억	이상미디어	20000	수정 삭제
8	야구를 부탁해	이상미디어	13000	수정 삭제
9	올림픽 이야기	삼성당	7500	수정 삭제
10	Olympic Champions	Pearson	13000	수정 삭제
103	HTML/CSS	웹프로그래밍	300002	수정 삭제
105	테스트도서	테스트출판사	75000	수정 삭제

도서목록

[처음으로 도서등록](#)

도서번호	도서명	출판사	가격	관리
1	축구의 역사2	굿스포츠2	70000	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
7	야구의 추억	이상미디어	20000	수정 삭제
8	야구를 부탁해	이상미디어	13000	수정 삭제
9	올림픽 이야기	삼성당	7500	수정 삭제
10	Olympic Champions	Pearson	13000	수정 삭제
103	HTML/CSS	웹프로그래밍	300002	수정 삭제
105	테스트도서2	테스트출판사2	80000	수정 삭제

4. 실행

12

고객 등록

고객등록

처음으로 고객목록

고객명	이순신
주소	충무
휴대폰	010-0000-0100
<input type="button" value="등록"/>	

고객 수정

고객수정

처음으로 고객목록

고객번호	15
고객명	장보고
주소	제주
휴대폰	010-1000-0100
<input type="button" value="수정"/>	

고객 삭제

13	손흥민	영국	011-0000-0002	수정 삭제
15	장보고	제주	010-1000-0100	수정 삭제

5	박세리	대한민국 부산		수정 삭제
13	손흥민	영국	011-0000-0002	수정 삭제

고객목록

처음으로 고객등록

고객번호	고객명	주소	휴대폰	관리
1	박지성	영국 맨체스터	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	박세리	대한민국 부산		수정 삭제
13	손흥민	영국	011-0000-0002	수정 삭제
15	이순신	충무	010-0000-0100	수정 삭제

고객목록

처음으로 고객등록

고객번호	고객명	주소	휴대폰	관리
1	박지성	영국 맨체스터	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	박세리	대한민국 부산		수정 삭제
13	손흥민	영국	011-0000-0002	수정 삭제
15	장보고	제주	010-1000-0100	수정 삭제