Spring DB

✔️ *JDBC*

✔️ *Spring Project 생성*

✔️ *MyBatis*

JDBC

> JDBC 이용하여 C R ***U*** D 처리

Class

CarRepository

```
void setUpdateCar(CarDTO car);
```

CarRepositoryImpl

```
@Override
public void setUpdateCar(CarDTO car) {
  if(car.getCfilename() != null) {//사진 변경 시
    String sql = "UPDATE car SET cname = ?, cprice = ?, ccate = ?, cdesc = ?, cfilename = ? WHERE cid = ?";

    template.update(sql, car.getCname(),
        car.getCprice(),
        car.getCcate(),
        car.getCdesc(),
        car.getCfilename(),
        car.getCid());
  } else if(car.getCfilename() == null) {//사진 변경 안할 시
    String sql = "UPDATE car SET cname = ?, cprice = ?, ccate = ?, cdesc = ? WHERE cid = ?";

    template.update(sql, car.getCname(),
        car.getCprice(),
        car.getCcate(),
        car.getCdesc(),
        car.getCid());
  }
}
```

CarService

```
void setUpdateCar(CarDTO car);
```

CarServiceImpl

```
@Override
public void setUpdateCar(CarDTO car) {
  carRepository.setUpdateCar(car);
}
```

CarController

```
@GetMapping("/update")
public String submitUpdateCar(@ModelAttribute("updateCar") CarDTO car, @RequestParam("id") String carId, Model model) {//객체, 조회, 보여

  CarDTO carById = carService.getCarById(carId);//조회
  model.addAttribute("car", carById);//보여주기

  return "updateForm";
}

@PostMapping("/update")
public String submitUpdateCar(@ModelAttribute("updateCar") CarDTO car) {

  MultipartFile carimage = car.getCarimage();//파일 처리

  String fname = carimage.getOriginalFilename();//파일명 처리
  File saveFile = new File(uploadPath + "/images", fname);

  if (carimage != null && !carimage.isEmpty()) {//이미지 없을 경우
    try {//이미지 있을 경우
      carimage.transferTo(saveFile);
      car.setCfilename(fname);
    } catch (Exception e) {
      throw new RuntimeException("차량 이미지 업로드가 실패했습니다.");
    }
  }

  carService.setUpdateCar(car);//DB에 넣기

  return "redirect:/cars";
}
```

View

product.jsp

```
<p><a href="<c:url value="/cars/update?id=${car.cid }"/>"
      class="btn btn-success btn-sm">수정</a><!-- 기본 (AJAX x) -->
```

updateCar.jsp

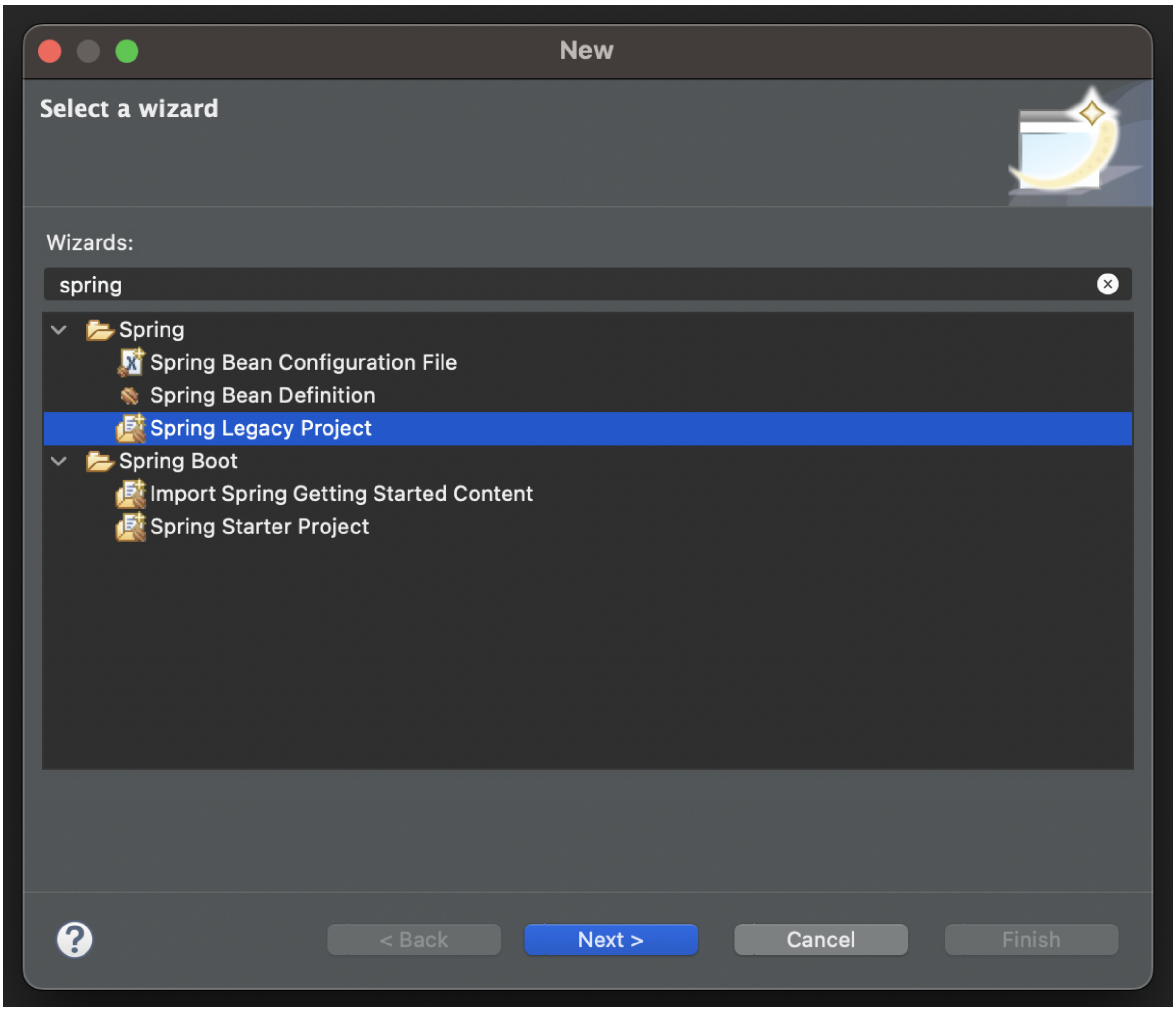
```
<form:form modelAttribute="updateCar"
  action="./update?${_csrf.parameterName}=${_csrf.token}"
  class="form-horizontal" method="post" enctype="multipart/form-data">
  <fieldset>
    <form:input path="cid" type="hidden" class="form-control" value="${car.cid }" />
    자동차 ID :
    ${car.cid }<br>
    자동차 이름 :
    <form:input path="cname" class="form-control" value="${car.cname }" />
    자동차 가격 :
    <form:input path="cprice" class="form-control" value="${car.cprice }" />
    자동차 카테고리 :
    <form:input path="ccate" class="form-control" value="${car.ccate }" />
    자동차 소개 :
    <textarea name="cdesc" cols="50" rows="2" class="form-control">${car.cdesc }</textarea>
    자동차 사진 :
    <form:input path="carimage" type="file" class="form-control" />
    <input type="submit" class="btn btn-primary" value="등록" />
  </fieldset>
</form:form>
```

tiles.xml

```
<definition name="updateCar" extends="base-Template">
    <put-attribute name="title" value="Update Car" />
    <put-attribute name="heading" value="정보 수정" />
    <put-attribute name="subheading" value="update Car" />
    <put-attribute name="content" value="/WEB-INF/views/updateCar.jsp" />
</definition>
```

Spring Project 생성

프로젝트 생성

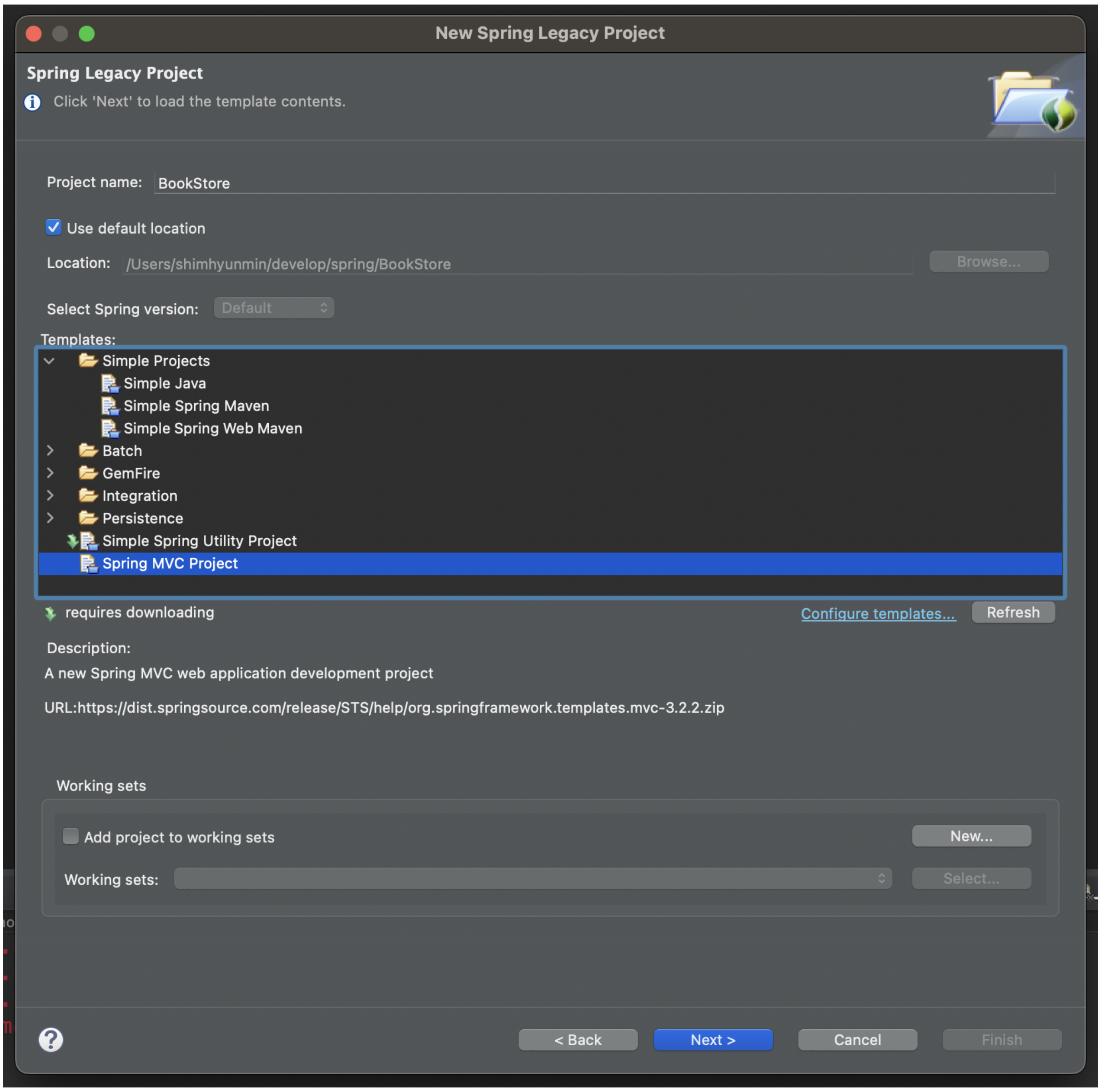

New

Select a wizard

Wizards:

spring

- Spring
  - Spring Bean Configuration File
  - Spring Bean Definition
  - Spring Legacy Project
- Spring Boot
  - Import Spring Getting Started Content
  - Spring Starter Project


New Spring Legacy Project

Spring Legacy Project

Click 'Next' to load the template contents.

Project name: BookStore

☑ Use default location

Location: /Users/shimhyunmin/develop/spring/BookStore

Select Spring version: Default

Templates:

- Simple Projects
  - Simple Java
  - Simple Spring Maven
  - Simple Spring Web Maven
- Batch
- GemFire
- Integration
- Persistence
- Simple Spring Utility Project
- Spring MVC Project

requires downloading

Description:

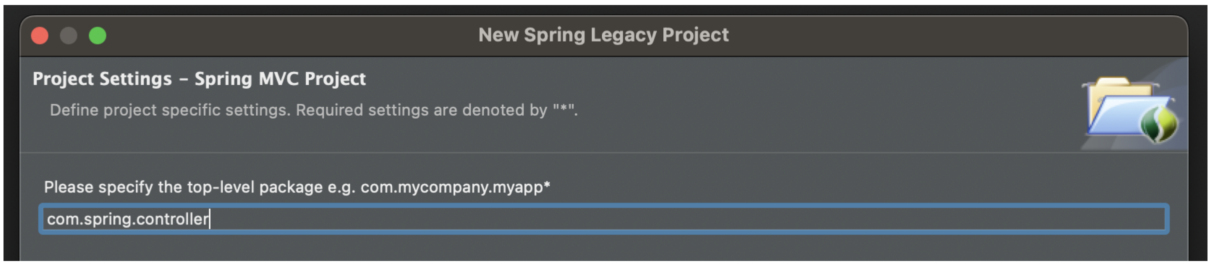
A new Spring MVC web application development project

URL:https://dist.springsource.com/release/STS/help/org.springframework.templates.mvc-3.2.2.zip

Working sets

☐ Add project to working sets

Working sets:


New Spring Legacy Project

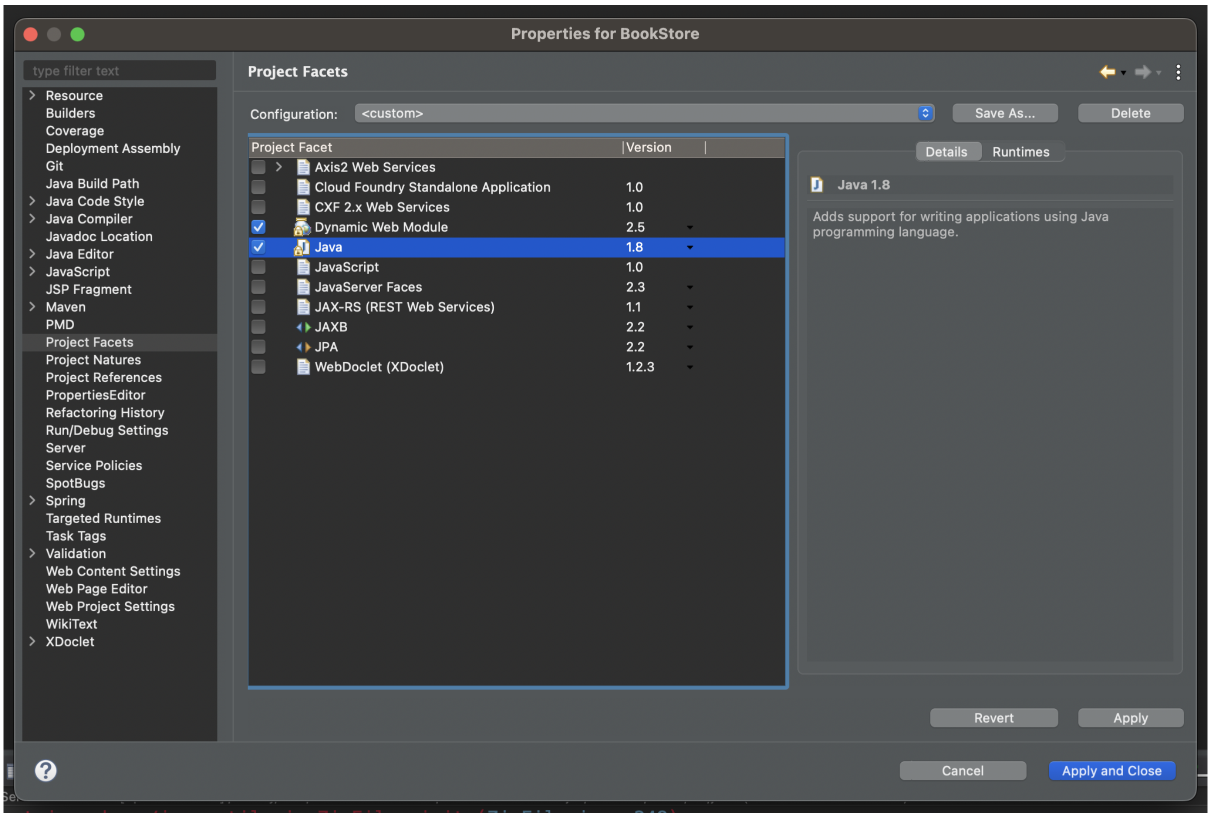
Project Settings – Spring MVC Project

Define project specific settings. Required settings are denoted by "*".

Please specify the top-level package e.g. com.mycompany.myapp*

com.spring.controller

자바 버전 변경


Properties for BookStore

Project Facets

Configuration: <custom>

Project Facet	Version
☐ Axis2 Web Services	
☐ Cloud Foundry Standalone Application	1.0
☐ CXF 2.x Web Services	1.0
✓ Dynamic Web Module	2.5
✓ Java	1.8
☐ JavaScript	1.0
☐ JavaServer Faces	2.3
☐ JAX-RS (REST Web Services)	1.1
☐ JAXB	2.2
☐ JPA	2.2
☐ WebDoclet (XDoclet)	1.2.3

Java 1.8

Adds support for writing applications using Java programming language.

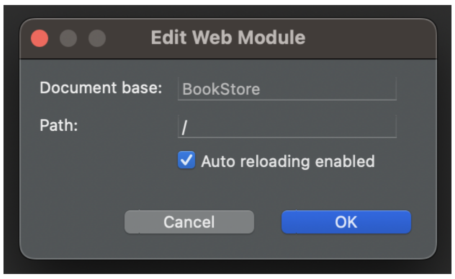
스프링 버전 변경

pom.xml

```
<org.springframework-version>5.3.19</org.springframework-version>
```

웹 경로 수정

server


Edit Web Module

Document base: BookStore

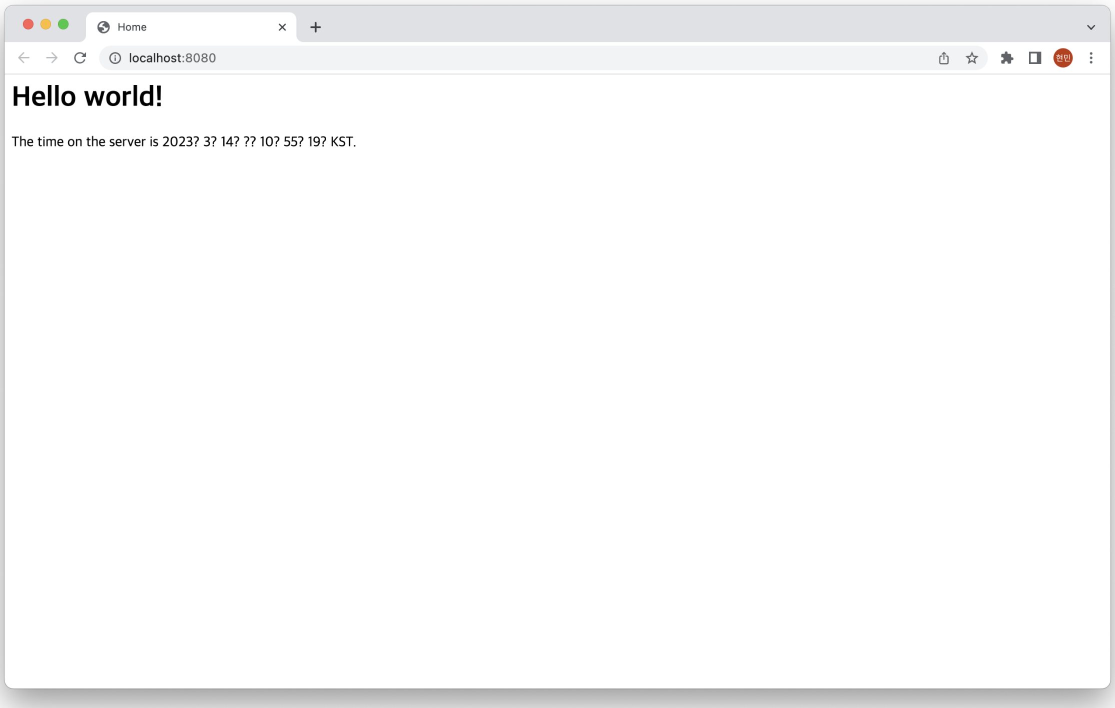
Path: /

✓ Auto reloading enabled

한글처리

web.xml

```
<filter>
  <filter-name>encodingFilter</filter-name>
  <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
  <init-param>
    <param-name>encoding</param-name>
    <param-value>UTF-8</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>encodingFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```



Hello world!

The time on the server is 2023? 3? 14? ?? 10? 55? 19? KST.

Class

BookController

@Controller	요청을 받아들이는 컨트롤러로 인지하여 자바 빈으로 등록하여 관리
ModelAndView	• 모델과 화면을 담당하는 뷰를 합친 객체 • 생성자를 통해 뷰의 경로 지정 가능

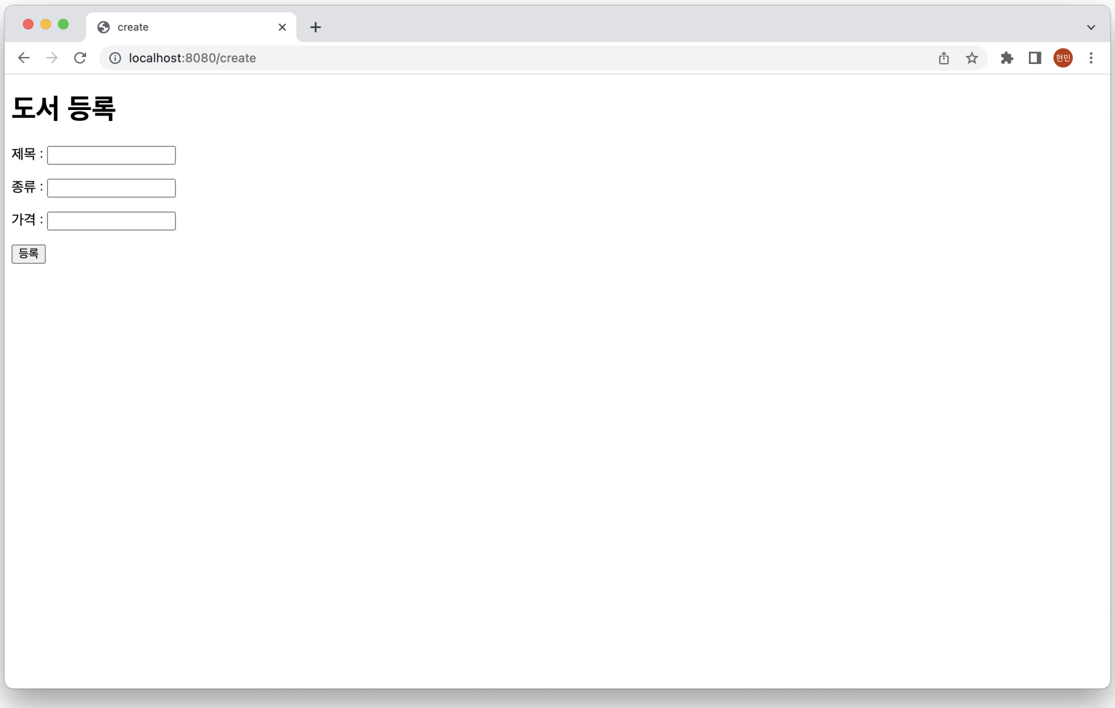
```
@Controller
public class BookController {

  @RequestMapping("/")
  public ModelAndView main() {
    return new ModelAndView("book/create");
  }

  @RequestMapping("/create")//경로
  public ModelAndView create() {//메소드 속성은 http 요청 메소드 의미
    return new ModelAndView("book/create");//jsp
  }
}
```

View

create.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
  pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>create</title>
</head>
```

```
<body>
  <h1>도서 등록</h1>
  <form method="post">
    <p>
      제목 : <input type="text" name="title" />
    <p>
      종류 : <input type="text" name="category" />
    <p>
      가격 : <input type="text" name="price" />
    <p>
      <input type="submit" value="등록" />
  </form>
</body>
</html>
```



도서 등록

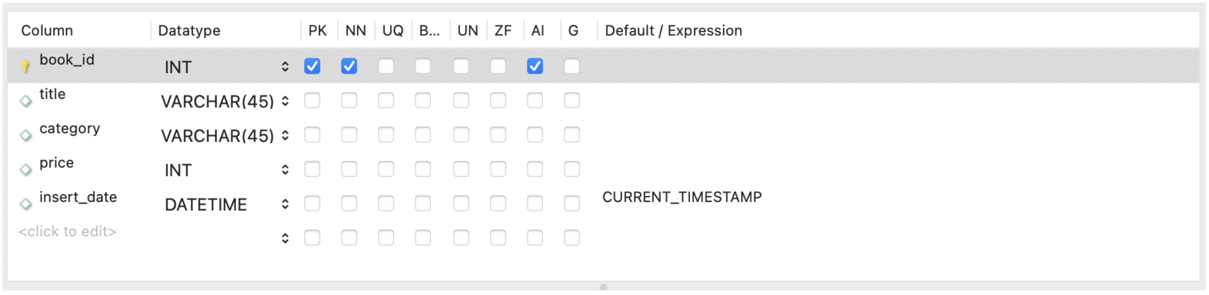
제목 :

종류 :

가격 :

등록

Table 생성



Column	Datatype	PK	NN	UQ	B...	UN	ZF	AI	G	Default / Expression
book_id	INT	✓	✓					✓		
title	VARCHAR(45)									
category	VARCHAR(45)									
price	INT									
insert_date	DATETIME									CURRENT_TIMESTAMP
<click to edit>										

MyBatis

> MyBatis 이용하여 C R U D 처리

의존성 주입

MyBatis Spring » 2.0.4

An easy-to-use Spring bridge for MyBatis sql mapping framework.

MyBatis » 3.5.6

The MyBatis SQL mapper framework makes it easier to use a relational database with object-oriented applications. MyBatis couples objects with stored procedures or SQL statements using a XML descriptor or annotations. Simplicity is the biggest advantage of the MyBatis data mapper over object relational mapping tools.

Spring JDBC » 5.3.19

Spring JDBC provides an abstraction layer that simplifies code to use JDBC and the parsing of database-vendor specific error codes.

Apache Commons DBCP » 2.7.0

Apache Commons DBCP software implements Database Connection Pooling

Log4Jdbc Log4j2 JDBC 4 » 1.16

Log4Jdbc Log4j2 JDBC 4

MySQL Connector Java » 8.0.28

MySQL Connector/J is a JDBC Type 4 driver, which means that it is pure Java implementation of the MySQL protocol and does not rely on the MySQL client libraries. This driver supports auto-registration with the Driver Manager, standardized validity checks, categorized SQLExceptions, support for large update counts, support for local and offset date-time variants from the java.time package, support for JDBC-4.x XML processing, support for per connection client information and support for the NCHAR, NVARCHAR ...

pom.xml

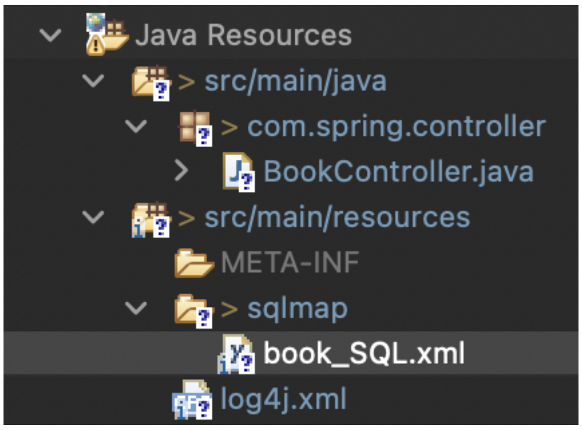
```
<!-- mybatis -->
<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis</artifactId>
  <version>3.5.6</version>
</dependency>

<!-- mybatis-spring -->
<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis-spring</artifactId>
  <version>2.0.4</version>
</dependency>

<!-- spring-jdbc -->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-jdbc</artifactId>
  <version>5.3.19</version>
</dependency>

<!-- commons-dbcp2 -->
<dependency>
  <groupId>org.apache.commons</groupId>
  <artifactId>commons-dbcp2</artifactId>
  <version>2.7.0</version>
</dependency>

<!-- log4jdbc-log4j2-jdbc4 -->
<dependency>
  <groupId>org.bgee.log4jdbc-log4j2</groupId>
  <artifactId>log4jdbc-log4j2-jdbc4</artifactId>
  <version>1.16</version>
</dependency>

<!-- mysql-connector-java -->
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>8.0.28</version>
</dependency>
```

DB 연결

root-context.xml

```
<!-- mysql 연결 설정 -->
<bean id="dataSource"
  class="org.springframework.jdbc.datasource.DriverManagerDataSource">
  <property name="driverClassName"
    value="com.mysql.cj.jdbc.Driver" />
  <property name="url"
    value="jdbc:mysql://localhost:3306/#?serverTimezone=UTC" />
  <property name="username" value="#" />
  <property name="password" value="#" />
</bean>
<bean id="sqlSessionFactory"
  class="org.mybatis.spring.SqlSessionFactoryBean">
  <property name="dataSource" ref="dataSource" />
  <property name="mapperLocations"
    value="classpath:/sqlmap/**/*_SQL.xml" />
</bean>
<bean id="sqlSessionTemplate"
  class="org.mybatis.spring.SqlSessionTemplate">
  <constructor-arg index="0" ref="sqlSessionFactory" />
</bean>
```

Mapper



book_SQL.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE  mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="book">
  <insert id="insert" parameterType="hashMap"
    useGeneratedKeys="true" keyProperty="book_id">
    <![CDATA[
      insert into book
      (title, category, price)
      values
      (#{title}, #{category}, #{price})
    ]]>
  </insert>
</mapper>
```

<insert></insert>	insert 구문
id=”insert”	쿼리문 이름
parameterType	• 데이터 형태 정의 • 객체를 받아서 사용 가능
useGeneratedKeys & keyProperty	useGeneratedKeys=”true”로 설정되면 쿼리 실행 후 생성된 PK를 keyProperty 속성에 넣어준다

Class

BookRepository

```
package com.spring.controller;

import java.util.Map;

public interface BookRepository {

  int insert(Map<String, Object> map);
}
```

BookRepositoryImpl

```
package com.spring.controller;

import java.util.Map;

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

@Repository
public class BookRepositoryImpl implements BookRepository {

  @Autowired
  SqlSessionTemplate sqlSessionTemplate;

  @Override
  public int insert(Map<String, Object> map) {

    return this.sqlSessionTemplate.insert("book.insert", map);
  }

}
```

BookService

연결만 해준다

```
package com.spring.controller;

import java.util.Map;

public interface BookService {

  String create(Map<String, Object> map);
}
```

BookServiceImpl

```
package com.spring.controller;

import java.util.Map;

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

@Service
public class BookServiceImpl implements BookService {

  @Autowired
  BookRepository bookRepository;
  @Override
  public String create(Map<String, Object> map) {

    int affectRowCount = this.bookRepository.insert(map);

    if(affectRowCount==1) {//insert 성공시 1, 실패시 0
      return map.get("book_id").toString();
    }
    return null;
  }

}
```

BookController

```
package com.spring.controller;

import java.util.Map;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.servlet.ModelAndView;

@Controller
public class BookController {

  @Autowired//의존성 주입
  BookService bookService;

  @GetMapping("/create")
  public ModelAndView createMethod() {

    return new ModelAndView("book/create");
  }

  @PostMapping("/create")//경로
  public ModelAndView create(@RequestParam Map<String, Object> map) {//메소드 속성은 http 요청 메소드 의미

    ModelAndView mav = new ModelAndView();

    String bookId = this.bookService.create(map);

    if(bookId==null) {
      mav.setViewName("redirect:/create");
    } else {
      //mav.setViewName("redirect:/detail?bookId=" + bookId);//detail 아직
      mav.setViewName("redirect:/create");
    }
    return mav;
  }
}
```