

# Spring Day 11

## ▼ Spring

### ▼ JDBC(UPDATE)

#### CarRepository.java

```
void setUpdateCar(CardTO car);
```

#### CarRepositoryImpl.java

```
@Override
public void setUpdateCar(CardTO car) {

    String sql = "UPDATE car SET cname=?, cprice=?, ccate=?, cdesc=?, cfilename=ifnull(?, cfilename) where cid=?";

    template.batchUpdate(sql, car.getCName(),
        car.getCPrice(),
        car.getCcate(),
        car.getCdesc(),
        car.getCFilename(),
        car.getCid());
}
```

#### CarService.java

```
void setUpdateCar(CardTO car);
```

#### CarServiceImpl.java

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

#### CarController.java

```
@GetMapping("/update")
public String requestUpdateCarForm(@RequestParam("id") String carId, Model model) {

    CardTO carById = carService.getCarById(carId);
    model.addAttribute("updateCar", carById);

    return "update";
}

@PostMapping("/update")
public String submitUpdateCar(@ModelAttribute("updateCar") CardTO car, HttpServletRequest request) {

    MultipartFile carimage = car.getCarimage();
    String fileName = carimage.getOriginalFilename();
}
```

```

File saveFile = new File(uploadPath + "\\images", fileName);

if (carimage != null && !carimage.isEmpty()) {
    try {
        carimage.transferTo(saveFile);
        car.setCfilename(fileName);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

carService.setUpdateCar(car);
return "redirect:/cars";
}

```

## update.jsp

```

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form" %>

<head>
<title>자동차 수정</title>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<meta charset="utf-8">

</head>
<body class="text-center">
<div class="alert alert-dark" role="alert">
<div class="container"><h1>차량 수정</h1>

    <form:form modelAttribute="updateCar" action="./update?${_csrf.parameterName}=${_csrf.token}" class="form-horizontal" enc

        <fieldset>
            <legend>
                차량 수정
            </legend>
            <form:input path="cid" type="hidden" class="form-control" value="${updateCar.cid}" />
                자동차 ID : ${updateCar.cid}<br>

                자동차 이름 :
                <form:input path="cname" class="form-control" value="${updateCar.cname}" />
                자동차 가격 :
                <form:input path="cprice" class="form-control" value="${updateCar.cprice}" />
                자동차 카테고리 :
                <form:input path="ccate" class="form-control" value="${updateCar.ccate}" />
                자동차 소개 :
                <textarea name="cdesc" cols="50" rows="2" class="form-control" >${updateCar.cdesc}</textarea>
                자동차 사진 :
                <form:input path="carimage" type="file" class="form-control" />

                <input type="submit" class="btn btn-primary" value="수정">

        </fieldset>
    </form:form>

</div>
</div>


</body>
</html>

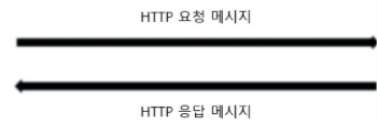
```

## ▼ @RequestBody & @ResponseBody

[Spring] @RequestBody / @ResponseBody 어노테이션이란?

스프링에서 비동기 처리를 하는 경우 @RequestBody, @ResponseBody를 사용한다. 비동기 처리를 위해 이 어노테이션들은 어떻게 작동할까? 클라이언트와 서버의 비동기 통신 클라이언트에서 서버로 통신하는 메시지를 요청(request) 메시지라고 하며, 서버에서 클라이언트로 통신하는 메시지를 응답(response) 메시지라고

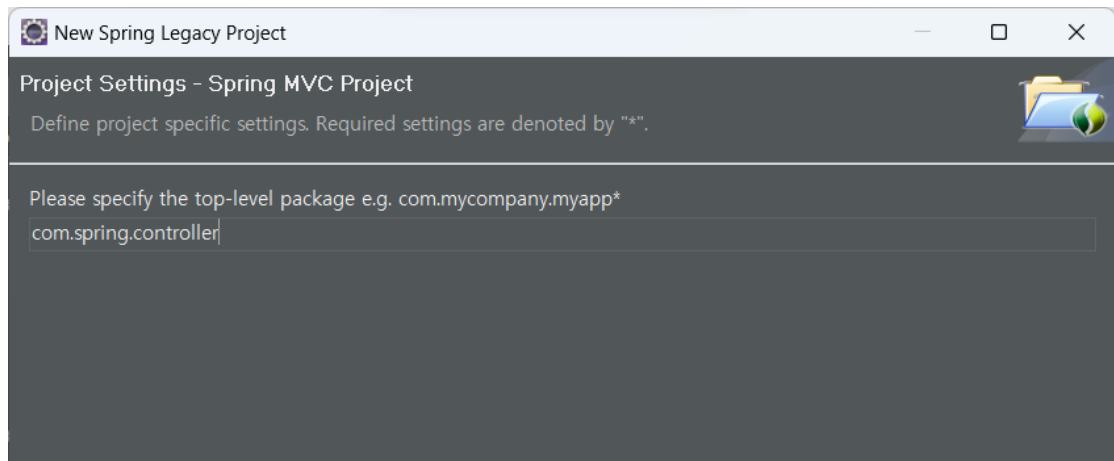
 <https://cheershennah.tistory.com/179>



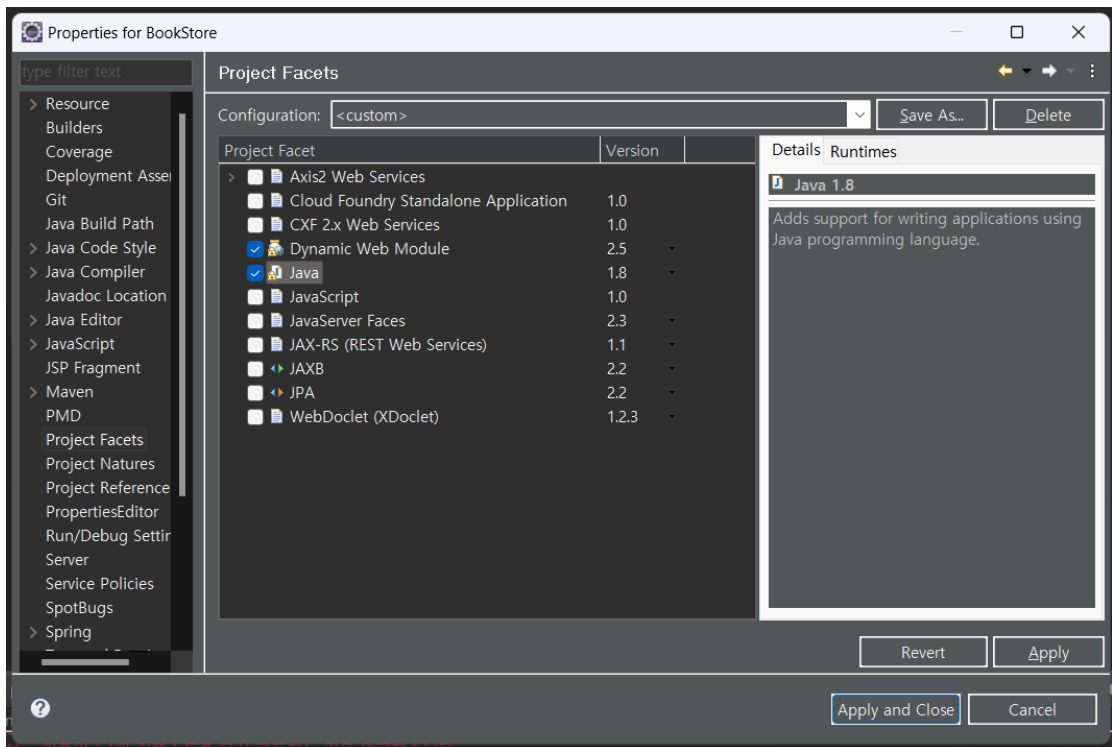
## ▼ MyBatis

### ▼ 환경 설정

#### Project Settings



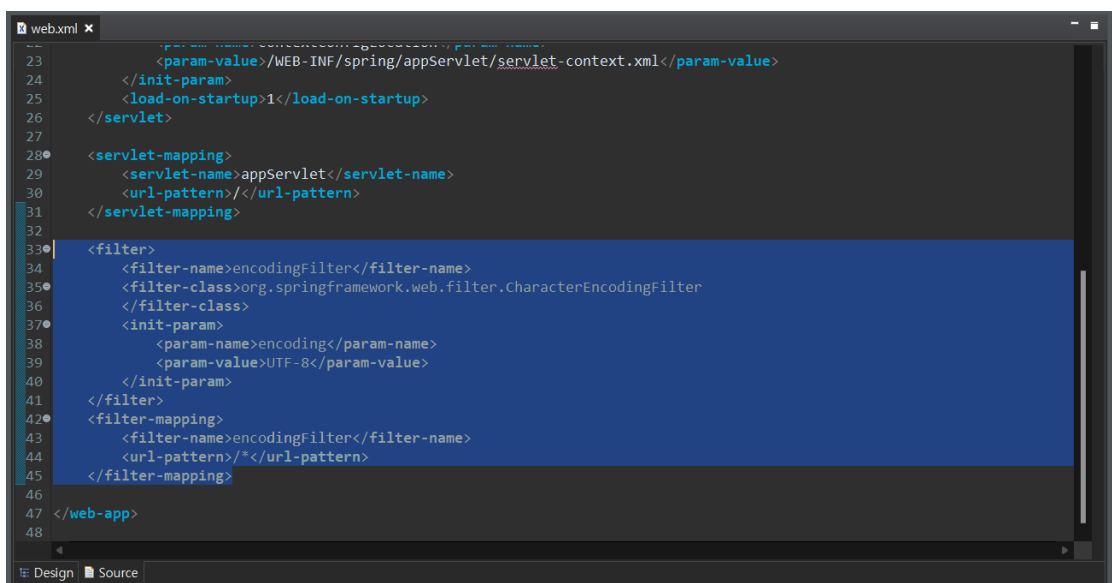
#### Java 1.8로 변경



## pom.xml 설정

```
<properties>
  <java-version>1.6</java-version>
  <org.springframework-version>5.3.19</org.springframework-version>
  <org.aspectj-version>1.6.10</org.aspectj-version>
  <org.slf4j-version>1.6.6</org.slf4j-version>
</properties>
```

## web.xml 설정(한글 인코딩)



## BookController.java

```
package com.spring.controller;

// ctrl + shift + o : import 자동 정리 기능.
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;

/**@Controller : Spring에서 Controller로 인식(Java Bean으로 등록하여 관리), Model 객체를 만들어 데이터를 담고 View를 반환.
@Controller
public class BookController {

    //ModelAndView : Model 데이터와 이동하고자 하는 View Page를 같이 저장한다.
    @RequestMapping("/")
    public ModelAndView main() {
        return new ModelAndView("book/create");
    }


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

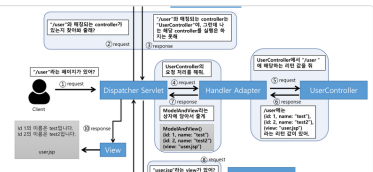
}
```

## @Controller & @RestController

### [Spring] @Controller와 @RestController의 차이점 알아보기

@Controller 와 @RestController Spring에서 컨트롤러를 지정해주기 위한 어노테이션은 @Controller와 @RestController가 있습니다. 전통적인 Spring MVC 컨트롤러인 @Controller와 RESTful 웹 서비스의 컨트롤러인 @RestController의 주요한 차이점은 HTTP Response Body가 생성되는 방식입니다. @Controller의

 <https://dev-coco.tistory.com/84>



## Model & ModelAndView

### [Spring] ModelAndView

Model과 차이점은 Model은 데이터만 저장하는데, ModelAndView는 데이터와 이동하고자 하는 View Page를 같이 저장한다. Controller 처리 결과 후 응답할 view와 view에 전달할 값을 저장생성자 ModelAndView(String viewName,

 <https://velog.io/@modsiw/Spring-ModelAndView>



## DB 생성

기본
옵션
인덱스 (1)
외래 키 (0)
제약 조건 확인 (0)
분할
CREATE 코드
ALTER 코드

이름:
book

코멘트:


열:
추가
제거
위로
아래로

#	이름	데이터 유형	길이/설정	부호 없...	NULL 허...	0으로...	기본값	코멘트	조합	표현식	가상
1	book_id	INT	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	AUTO_INCREMENT...				
2	title	VARCHAR	50	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL		utf8mb4_0900_ai_ci		
3	category	VARCHAR	50	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL		utf8mb4_0900_ai_ci		
4	price	INT	10	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL				
5	insert_date	DATETIME		<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	CURRENT_TIMESTAMP...				

```
CREATE TABLE `book` (
  `book_id` INT(10) NOT NULL AUTO_INCREMENT,
  `title` VARCHAR(50) NULL DEFAULT NULL COLLATE 'utf8mb4_0900_ai_ci',
  `category` VARCHAR(50) NULL DEFAULT NULL COLLATE 'utf8mb4_0900_ai_ci',
  `price` INT(10) NULL DEFAULT NULL,
  `insert_date` DATETIME NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`book_id`) USING BTREE
)
COLLATE='utf8mb4_0900_ai_ci'
ENGINE=InnoDB
;
```

pom.xml 설정(의존성 주입)

Home » org.mybatis » mybatis » 3.5.6



**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.

License	Apache 2.0
Categories	Object/Relational Mapping
Tags	<p>           persistence           relational           mapping </p>
HomePage	http://www.mybatis.org/mybatis-3
Date	Oct 06, 2020
Files	<p>           jar (1.7 MB)           View All </p>
Repositories	Central
Ranking	#313 in MvnRepository (See Top Artifacts) #3 in Object/Relational Mapping
Used By	1,422 artifacts
Vulnerabilities	Vulnerabilities from dependencies: CVE-2022-45868 CVE-2022-41946 CVE-2022-41853 View 17 more ...



## MyBatis Spring » 2.0.4

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

License	Apache 2.0
Tags	persistence spring
HomePage	<a href="http://www.mybatis.org/spring/">http://www.mybatis.org/spring/</a>
Date	Mar 08, 2020
Files	<a href="#">jar (66 KB)</a> <a href="#">View All</a>
Repositories	Central Mulesoft
Ranking	#708 in MvnRepository (See Top Artifacts)
Used By	615 artifacts
Vulnerabilities	<b>Vulnerabilities from dependencies:</b> <a href="#">CVE-2022-41853</a> <a href="#">CVE-2022-23307</a> <a href="#">CVE-2022-23305</a> <a href="#">View 6 more ...</a>

**Note:** There is a new version for this artifact

New Version	3.0.1
-------------	-------



## 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.

License	Apache 2.0
Categories	JDBC Extensions
Tags	sql jdbc spring
Organization	Spring IO
HomePage	<a href="https://github.com/spring-projects/spring-framework">https://github.com/spring-projects/spring-framework</a>
Date	Apr 13, 2022
Files	<a href="#">pom (2 KB)</a> <a href="#">jar (418 KB)</a> <a href="#">View All</a>
Repositories	Central Spring Releases USIT
Ranking	#111 in MvnRepository (See Top Artifacts) #1 in JDBC Extensions
Used By	4,015 artifacts
Vulnerabilities	<b>Vulnerabilities from dependencies:</b> <a href="#">CVE-2022-22971</a> <a href="#">CVE-2022-22970</a>

**Note:** There is a new version for this artifact

New Version	6.0.6
-------------	-------

Home » [org.apache.commons](#) » [commons-dbcp2](#) » 2.7.0



## Apache Commons DBCP » 2.7.0

Apache Commons DBCP software implements Database Connection Pooling

License	<a href="#">Apache 2.0</a>
Categories	<a href="#">JDBC Pools</a>
Tags	<a href="#">sql</a> <a href="#">jdbc</a> <a href="#">apache</a> <a href="#">pool</a> <a href="#">commons</a>
HomePage	<a href="https://commons.apache.org/dbcp/">https://commons.apache.org/dbcp/</a>
Date	Aug 07, 2019
Files	<a href="#">jar (203 KB)</a> <a href="#">View All</a>
Repositories	<a href="#">Central</a>
Ranking	#483 in <a href="#">MvnRepository</a> (See Top Artifacts) #3 in <a href="#">JDBC Pools</a>
Used By	<a href="#">897 artifacts</a>
Vulnerabilities	<b>Vulnerabilities from dependencies:</b> <a href="#">CVE-2022-45868</a> <a href="#">CVE-2022-23221</a> <a href="#">CVE-2021-42392</a> <a href="#">View 1 more ...</a>

**Note:** There is a new version for this artifact

New Version	<a href="#">2.9.0</a>
-------------	-----------------------

Home » [org.bgee.log4jdbc-log4j2](#) » [log4jdbc-log4j2-jdbc4](#) » 1.16



## Log4Jdbc Log4j2 JDBC 4 » 1.16

Log4Jdbc Log4j2 JDBC 4

License	<a href="#">Apache 2.0</a>
Tags	<a href="#">sql</a> <a href="#">logging</a> <a href="#">jdbc</a>
Date	Dec 12, 2013
Files	<a href="#">pom (2 KB)</a> <a href="#">jar (89 KB)</a> <a href="#">View All</a>
Repositories	<a href="#">Central</a> <a href="#">Sonatype</a> <a href="#">WSO2 Public</a>
Ranking	#39951 in <a href="#">MvnRepository</a> (See Top Artifacts)
Used By	<a href="#">9 artifacts</a>

[Maven](#) [Gradle](#) [Gradle \(Short\)](#) [Gradle \(Kotlin\)](#) [SBT](#) [Ivy](#) [Grape](#) [Leiningen](#) [Buildr](#)

```
<!-- https://mvnrepository.com/artifact/org.bgee.log4jdbc-log4j2/log4jdbc-log4j2-jdbc4 -->
<dependency>
  <groupId>org.bgee.log4jdbc-log4j2</groupId>
  <artifactId>log4jdbc-log4j2-jdbc4</artifactId>
  <version>1.16</version>
</dependency>
```

☒ Include comment with link to declaration





## 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 ...

License	GPL 2.0
Categories	JDBC Drivers
Tags	database sql jdbc driver connector mysql
Organization	Oracle Corporation
HomePage	<a href="http://dev.mysql.com/doc/connector-j/en/">http://dev.mysql.com/doc/connector-j/en/</a>
Date	Jan 17, 2022
Files	<a href="#">pom (2 KB)</a> <a href="#">jar (2.4 MB)</a> <a href="#">View All</a>
Repositories	Central
Ranking	#68 in MvnRepository (See Top Artifacts) #1 in JDBC Drivers
Used By	6,903 artifacts
Vulnerabilities	<b>Vulnerabilities from dependencies:</b> <a href="#">CVE-2022-3510</a> <a href="#">CVE-2022-3509</a> <a href="#">CVE-2022-3171</a> <a href="#">View 1 more ...</a>

```

<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis -->
<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis</artifactId>
  <version>3.5.6</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.mybatis/mybatis-spring -->
<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis-spring</artifactId>
  <version>2.0.4</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-jdbc</artifactId>
  <version>5.3.19</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.apache.commons/commons-dbcp2 -->
<dependency>
  <groupId>org.apache.commons</groupId>
  <artifactId>commons-dbcp2</artifactId>
  <version>2.7.0</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.bgee.log4jdbc-log4j2/log4jdbc-log4j2-jdbc4 -->
<dependency>
  <groupId>org.bgee.log4jdbc-log4j2</groupId>
  <artifactId>log4jdbc-log4j2-jdbc4</artifactId>
  <version>1.16</version>
</dependency>
<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>8.0.28</version>
</dependency>

```

### servlet-context.xml 설정

```

<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/mvc"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:beans="http://www.springframework.org/schema/beans"

```

```

xmlns:context="http://www.springframework.org/schema/context"
xsi:schemaLocation="http://www.springframework.org/schema/mvc https://www.springframework.org/schema/mvc/spring-mvc.xsd
http://www.springframework.org/schema/beans https://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context https://www.springframework.org/schema/context/spring-context.xsd">

<!-- DispatcherServlet Context: defines this servlet's request-processing infrastructure -->

<!-- Enables the Spring MVC @Controller programming model -->
<annotation-driven />

<!-- Handles HTTP GET requests for /resources/** by efficiently serving up static resources in the ${webappRoot}/resources directory -->
<resources mapping="/resources/**" location="/resources/" />

<!-- Resolves views selected for rendering by @Controllers to .jsp resources in the /WEB-INF/views directory -->
<beans:bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
  <beans:property name="prefix" value="/WEB-INF/views/" />
  <beans:property name="suffix" value=".jsp" />
</beans:bean>

<context:component-scan base-package="com.spring.*" />

</beans:beans>

```

## Bean 설정이 중요한 이유

[스프링] No qualifying bean of type " available: expected at least 1 bean which qualifies as autowire

Service 인터페이스를 Implements 하는 클래스에서 @Service 어노테이션이 빠지지 않았는지 체크해보자....

<https://m.blog.naver.com/mmmmsolzer/222010770813>



## root-context.xml 설정

```

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.springframework.org/schema/beans https://www.springframework.org/schema/beans/spring-beans.xsd"

<!-- Root Context: defines shared resources visible to all other web components -->

<!-- mysql 연결 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/difbfl4751?serverTimezone=UTC"/>
  <property name="username" value="difbfl4751"/>
  <property name="password" value="비밀번호"/>
</bean>

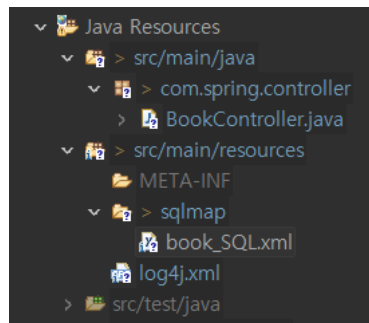
<bean id="sqlSessionFactory"
class="org.mybatis.spring.SqlSessionFactoryBean">
  <property name="dataSource" ref="dataSource" />
  <property name="mapperLocations"
value="classpath:/sqlmap/**/*.xml" />
</bean>

<bean id="sqlSessionTemplate"
class="org.mybatis.spring.SqlSessionTemplate">
  <constructor-arg index="0" ref="sqlSessionFactory" />
</bean>

</beans>

```

## Mapper 위치



## Mapper 설정

```
<?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 : MyBatis의 데이터 입력 태그 -->
<!-- id : insert 태그의 id -->
<!-- parameterType : 데이터의 형태 -->
<!-- useGeneratedKeys & keyProperty : useGeneratedKeys를 true로 설정하면 MyBatis에서 insert 쿼리 실행 후 생성된 PK를 keyProperty 속성에 저장 -->
<insert id="insert" parameterType="hashMap" useGeneratedKeys="true" keyProperty="book_id" >
<![CDATA[
    insert into book
    (title, category, price)
    values
    ({title}, #{category}, #{price})
]]>
</insert>

</mapper>
```

## BookRepository.java

```
package com.spring.repository;

import java.util.Map;

public interface BookRepository {

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

}
```

## BookRepositoryImpl.java

```
package com.spring.repository;

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.java

```
package com.spring.service;

import java.util.Map;

public interface BookService {

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

}
```

### BookServiceImpl.java

```
package com.spring.service;

import java.util.Map;

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

import com.spring.repository.BookRepository;

@Service
public class BookServiceImpl implements BookService {

    @Autowired
    BookRepository bookRepository;

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

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

        // insert 구문은 입력이 성공하면 1, 실패하면 0을 반환한다.
        if(affectRowCount == 1) {
            return map.get("book_id").toString();
        }

        return null;
    }

}
```

### BookController.java

```
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;
// ctrl + shift + o : import 자동 정리 기능.
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.servlet.ModelAndView;

import com.spring.service.BookService;

/**@Controller : Spring에서 Controller로 인식(Java Bean으로 등록하여 관리), Model 객체를 만들어 데이터를 담고 View를 반환.
@Controller
public class BookController {

    @Autowired
    BookService bookService;
```

```

//ModelAndView : Model 데이터와 이동하고자 하는 View Page를 같이 저장한다.
@RequestMapping("/")
public ModelAndView main() {
    return new ModelAndView("book/create");
}

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

@RequestMapping(value="/create", method=RequestMethod.POST)
public ModelAndView create(@RequestParam Map<String, Object> map) {
    ModelAndView mav = new ModelAndView();

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

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

    return mav;
}
}

```

## create.jsp

```

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

</form>

</body>
</html>

```

## ▼ Python

### Pandas

#### pickle

```
import pandas as pd

scientists = pd.read_csv('scientists.csv')

# 데이터 저장하기
# 잘 가공한 데이터를 안전하게 보관해야 다음에 또 쉽게 사용할 수 있다.

# 1. 피클로 저장하기
# 피클은 데이터를 바이너리 형태로 직렬화한 오브젝트이다. 피클로 저장하면 용량을 아주 작게 저장할 수 있어 편리하다.

names = scientists['Name']
names

names.to_pickle('sci_names.pickle')

scientists.to_pickle('sci_all.pickle')
# 데이터프레임 전체를 피클로 저장할 수 있다.
# 피클로 저장하면 오브젝트이기 때문에 일반 편집기같은 프로그램으로는 열기 어렵다.

# 1. 피클 읽어오기

sci_names_from_pickle = pd.read_pickle('sci_names.pickle')
sci_names_from_pickle

sci_df_from_pickle = pd.read_pickle('sci_all.pickle')
sci_df_from_pickle

# CSV, TSV 로 저장하기

# CSV는 데이터를 싼표/콜마 로 구분하여 저장한 파일
# TSV는 데이터를 탭으로 구분하여 저장한 파일
# 일반 프로그램으로 쉽게 열어서 볼 수 있다.

names

names.to_csv('sci_names_csv.csv')

# TSV 로 저장

names.to_csv('sci_names_csv.tsv', sep='\t')

# 엑셀로 저장

# !pip install openpyxl

import openpyxl

scientists.to_excel('scientists_df_xlsx.xlsx')
```

## 데이터 연결하기

### [Python] 데이터프레임 합치기 :: pd.merge()

pd.merge는 공통의 열을 기준으로 두 데이터프레임을 합쳐준다. sql에서 join과 같은 역할이다. import pandas as pd # 기준열 이름이 같을 때 pd.merge(left, right, on = '기준열', how = '조인방식') # 기준열 이름이 다를 때 pd.merge(left, right, left\_on = '왼쪽 열', right\_on = '오른쪽 열', how = '조인방식') left : 왼쪽 데이터프레임 right :

👉 <https://mizykk.tistory.com/82>



### 08-04. 객체병합 (merge)

```
####DataFrame.merge(right, how='inner', on=None, left_on=None, right_on=None, left_index=False, right_index=False)
```

👉 <https://wikidocs.net/153875>



```
# 데이터 연결하기

# 분석하기 좋은 데이터 = 분석하기 쉬운 데이터.
# 실전에서도 전체 데이터 분석작업의 80% 이상을 차지하는 것이 전처리 작업(pre processing)이다.

# 분석하기 좋은 데이터의 기본 조건
# 1. 변수는 열로 구성
# 2. 값은 행으로 구성
# 3. 열과 행으로 구성된 표 형태

df1 = pd.read_csv('concat_1.csv')
```

```

df2 = pd.read_csv('concat_2.csv')
df3 = pd.read_csv('concat_3.csv')

row_concat = pd.concat([df1, df2, df3])
print(row_concat)

print(row_concat.iloc[3, ])

new_row_series = pd.Series(['n1', 'n2', 'n3', 'n4'])

print(pd.concat([df1, new_row_series]))

# NaN 은 누락값/결측치이다. 시리즈가 새로운 열로 삽입 되었다.
# 시리즈를 데이터프레임에 넣으려고 하면 제대로 들어가지지 않는다.
# 왜냐면 열이름이 없기 때문이다.

# 데이터프레임에 추가하려면 반드시 데이터프레임에 담아서 연결해야 한다.

new_row_df = pd.DataFrame(['n1', 'n2', 'n3', 'n4'], columns=['A', 'B', 'C', 'D'])
print(new_row_df)

# concat

print(pd.concat([df1, new_row_df]))

#append

print(df1.append(new_row_df))

# append는 붙이는 방식이기 때문에 인덱스가 0으로 다시 시작한다.

# ignore_index=True : 기존 인덱스를 무력화 시키고 0부터 새롭게 인덱스 부여

data_dict = {'A' : 'n1', 'B' : 'n2', 'C' : 'n3', 'D' : 'n4',}
print(df1.append(data_dict, ignore_index=True))

# 열 방향으로 데이터 붙이기
col_concat = pd.concat([df1, df2, df3], axis=1)
print(col_concat)

# 같은 이름의 열들만 추출
print(col_concat['A'])

# 새로운 열을 간단히 추가

col_concat['new_col_list'] = ['n1', 'n2', 'n3', 'n4']
print(col_concat)

print( pd.concat([df1, df2, df3], axis=1, ignore_index=True))

# 공통 열과 공통 인덱스만 연결

df1.columns = ['A', 'B', 'C', 'D']
df2.columns = ['E', 'F', 'G', 'H']
df3.columns = ['A', 'C', 'F', 'H']

print(df1)
print(type(df1))

print(df2)
print(type(df2))

print(df3)
print(type(df3))

row_concat = pd.concat([df1, df2, df3])
print(row_concat)

# 데이터프레임의 공통 열들만 골라서 연결하면 NaN 이 생기지 않을 것이다.
# 그러나 df1, df2, df3 모두의 공통열은 존재하지 않는다.
# 만약에 공통열이 있다면 공통열만을 추출하는 명령은 join='inner'이다.

print(pd.concat([df1, df2, df3], join='inner'))

print(pd.concat([df1, df3], ignore_index=False, join='inner'))

# 행 방향으로 연결
df1.index = [0, 1, 2, 3]
df2.index = [4, 5, 6, 7]
df3.index = [0, 2, 5, 7]

print(df1)

col_concat = pd.concat([df1, df2, df3], axis=1)
print(col_concat)

# index를 통일해서 연결하기

```

```

# merge 매서드를 사용하여 데이터 연결하기

person = pd.read_csv('survey_person.csv')
site = pd.read_csv('survey_site.csv')
survey = pd.read_csv('survey_survey.csv')
visited = pd.read_csv('survey_visited.csv')

print(person)
print(site)

visited_subset = visited.loc[[0, 2, 6], ]
print(visited_subset)

o2o_merge = site.merge(visited_subset, left_on='name', right_on='site')
print(o2o_merge)

site

m2o_merge = site.merge(visited, left_on='name', right_on='site')
print(m2o_merge)

visited

ps = person.merge(survey, left_on='ident', right_on='person')
vs = visited.merge(survey, left_on='ident', right_on='taken')

print(ps)

```

## 전처리 작업(Pre Processing)

```

# 전처리 : NaN(누락값, 결측치) 처리
# NaN NAN nan 모두 가능하다. NaN을 선호한다.
from numpy import NAN, NaN, nan

# NaN는 0도 아니고 ''도 아니다. 말 그대로 값이 없음을 나타낸다.

# 따라서 '같다'도 사용할 수 없다.

print(NaN == True)
print(NaN == False)
print(NaN == 0)
print(NaN == '')

# NaN은 값 자체가 없기 때문에 자기 자신도 True 가 아니라 False 로 출력한다.
print(NaN == NaN)

print(NaN == nan)
print(NaN == NAN)
print(nan == NAN)

import pandas as pd

print(pd.isnull(NaN))
print(pd.isnull(nan))
print(pd.isnull(NAN))

# 원래 결측치가 존재하는 파일도 있지만
# 데이터를 연결하거나 입력하는 과정에서 결측치가 발생하기도 한다.

visited = pd.read_csv('survey_visited.csv')
survey = pd.read_csv('survey_survey.csv')

print(visited)

print(survey)

vs = visited.merge(survey, left_on='ident', right_on='taken')
print(vs)

# 결측치의 개수 구하기

# 전체 개수
ebola = pd.read_csv('country_timeseries.csv')
print(ebola.count())

num_rows = ebola.shape[0] # 행의 개수
num_missing = num_rows - ebola.count()
print(num_missing)

ebola.isnull()

```



```

ebola['Cases_Guinea'].isnull()

print(ebola.Cases_Guinea.value_counts(dropna=False).head()) # (dropna=False) : 결측값을 유일한 값에 포함할지 여부

print(ebola.Cases_Guinea.value_counts(dropna=False))

# 결측치 처리

# 1. 임의의 값을 넣는다(0, 평균값, 최빈값 등)
# 2. 삭제한다.

# 1. 임의의 값(0) 으로 대체하는 방법
print(ebola.fillna(0).iloc[0:10, 0:5])

# 2. 삭제한다.

print(ebola.shape)

ebola_dropna = ebola.dropna()
print(ebola_dropna.shape)

print(ebola_dropna)

# 데이터 분석과 인공지능 분야에서는 결측치를 처리하는 능력이 중요하다.
# 결측치를 처리하는 방법은 반드시 숙지해야 한다.

```

## 자료형 변환

### 판다스 입문(판다스 자료형)

#13 판다스 chapter 13. 판다스 자료형 판다스 자료형 다루기 - 자료형 변환하기 자료형 변환은 데이터 분석 과정에서 반드시 알아야 하는 요소중 하나이다. 예를 들어 카테고리형은 문자열로 변환해야 데이터 분석을 더 수월하게 할 수 있기 때문에 자주 변환하는 자료형이다. 하지만 전화번호로 평균을 구하거나 더하는 등의 계산은 거의  
 :: <https://speedanddirection.tistory.com/31>



```

# 자료형 변환

# !pip install seaborn

import seaborn as sns # matplotlib() 보다 그림을 조금 더 세련되게 그릴 수 있다.

tips = sns.load_dataset("tips")

tips

tips.dtypes

tips['sex_str'] = tips['sex'].astype(str)
print(tips.dtypes)

tips['total_bill'] = tips['total_bill'].astype(str)
print(tips.dtypes)

tips['total_bill'] = tips['total_bill'].astype(float)
print(tips.dtypes)

# 이상한 데이터 & 잘못된 데이터를 변환

# 정수가 있어야하는 열에 문자가 있으면 변환이 필요하다.
tips_sub_miss = tips.head(10)
tips_sub_miss

tips_sub_miss.loc[[1, 3, 5, 7], 'total_bill'] = 'missing'

tips_sub_miss

# 'total_bill' 데이터에 문자가 포함되어 있어서 자료형 자체가 변경되었다.

print(tips_sub_miss.dtypes)

# 'total_bill' 데이터에 문자가 포함되어 있어서 자료형 변환이 불가능하다.

# tips_sub_miss['total_bill'].astype(float)
# pd.to_numeric(tips_sub_miss['total_bill'])

# to_numeric 매서드의 errors 인자에 coerce 를 지정하면 숫자로 변환할 수 없는 값을 누락값으로 지정한다.

tips_sub_miss['total_bill'] = pd.to_numeric(tips_sub_miss['total_bill'], errors='coerce')

```

```
print(tips_sub_miss.dtypes)

# 판다스에는 카테고리(유한한 범위 값만 가질 수 있는 자료형)라는 특수한 자료형이 있다.
# 만약에 5개의 언어만 저장할 열이 있다고 생각한다면 문자열 자료형보다는
# 카테고리 자료형을 사용하는 것이 용량과 속도면에서도 훨씬 효율적이다.

# 카테고리 자료형
# 1. 용량과 속도면에서 탁월하다.
# 2. 주로 동일한 문자열이 반복되어 데이터를 구성하는 경우에 사용된다.

tips['sex'] = tips['sex'].astype('str')
print(tips.info())

tips['sex'] = tips['sex'].astype('category')
print(tips.info())

tips['sex'] = tips['sex'].astype('str')
tips['smoker'] = tips['smoker'].astype('str')
tips['day'] = tips['day'].astype('str')
tips['time'] = tips['time'].astype('str')
print(tips.info())
```

## groupby

### 판다스 입문(그룹연산 - 데이터 집계, 데이터 변환)

#16 판다스 chapter 16. 그룹연산 #1 그룹 연산 데이터 집계 데이터 집계하기 앞서 배웠던 갭마인더 데이터 집합으로 각 연도의 평균 수명을 구했던 것을 토대로 수집한 데이터를 바탕으로 평균이나 합 등을 구하여 의미있는 값을 도출해 내는 것을 '집계'라고 한다. 데이터를 집계하면 전체 데이터를 요약, 정리하여 볼 수 있기 때문에 데이터

👉 <https://speedanddirection.tistory.com/34>



```
#groupby

import pandas as pd
df = pd.read_csv('gapminder.tsv', sep='\t')

df

# 'lifeExp'의 평균값, 'pop'과 'gdpPercap'의 중위값 구하기

gdf_dict = df.groupby('year').agg({'lifeExp': 'mean', 'pop': 'median', 'gdpPercap': 'median'})
print(gdf_dict)
```