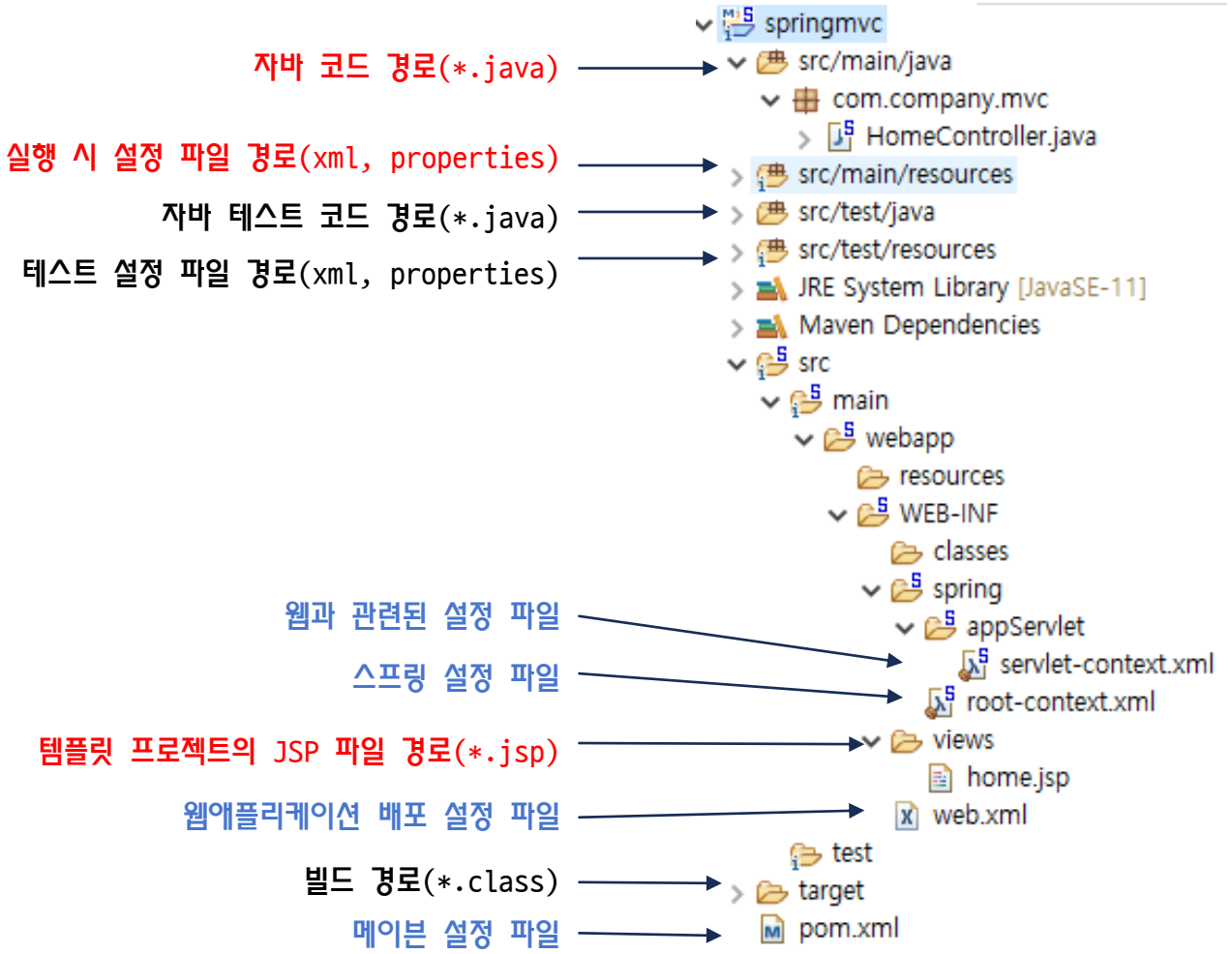
3. 스프링 MVC 프레임워크

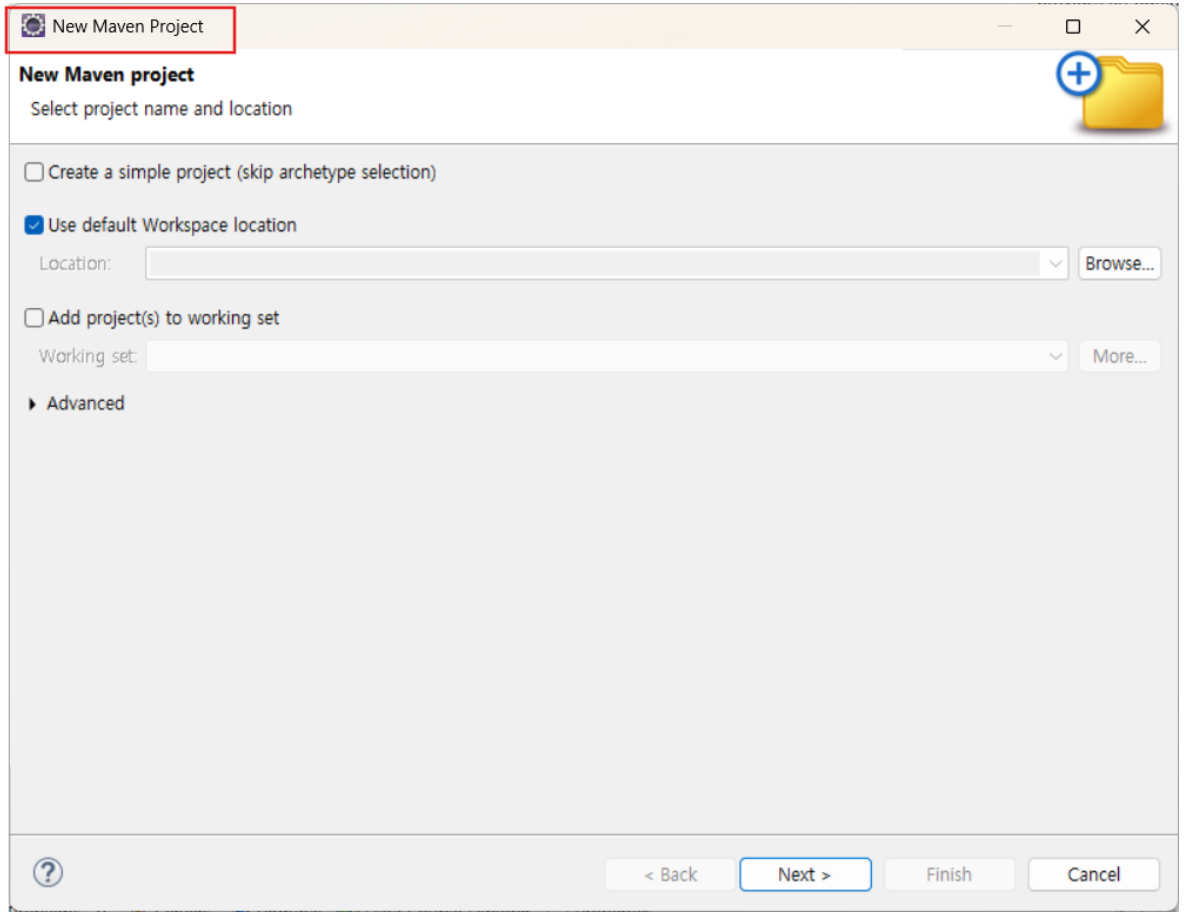
1. 스프링 MVC 프로젝트
2. Mybatis 설정
3. JUNIT
4. 로그 설정
5. 컨트롤러와 웹페이지 작성
6. DYNAMIC WEB PROJECT 를 SPRING 프로젝트로 변경

1.1 프로젝트 구조


자바 코드 경로(*.java)
실행 시 설정 파일 경로(xml, properties)
자바 테스트 코드 경로(*.java)
테스트 설정 파일 경로(xml, properties)
웹과 관련된 설정 파일
스프링 설정 파일
템플릿 프로젝트의 JSP 파일 경로(*.jsp)
웹애플리케이션 배포 설정 파일
빌드 경로(*.class)
메이븐 설정 파일
springmvc
src/main/java
com.company.mvc
HomeController.java
src/main/resources
src/test/java
src/test/resources
JRE System Library [JavaSE-11]
Maven Dependencies
src
main
webapp
resources
WEB-INF
classes
spring
appServlet
servlet-context.xml
root-context.xml
views
home.jsp
web.xml
test
target
pom.xml

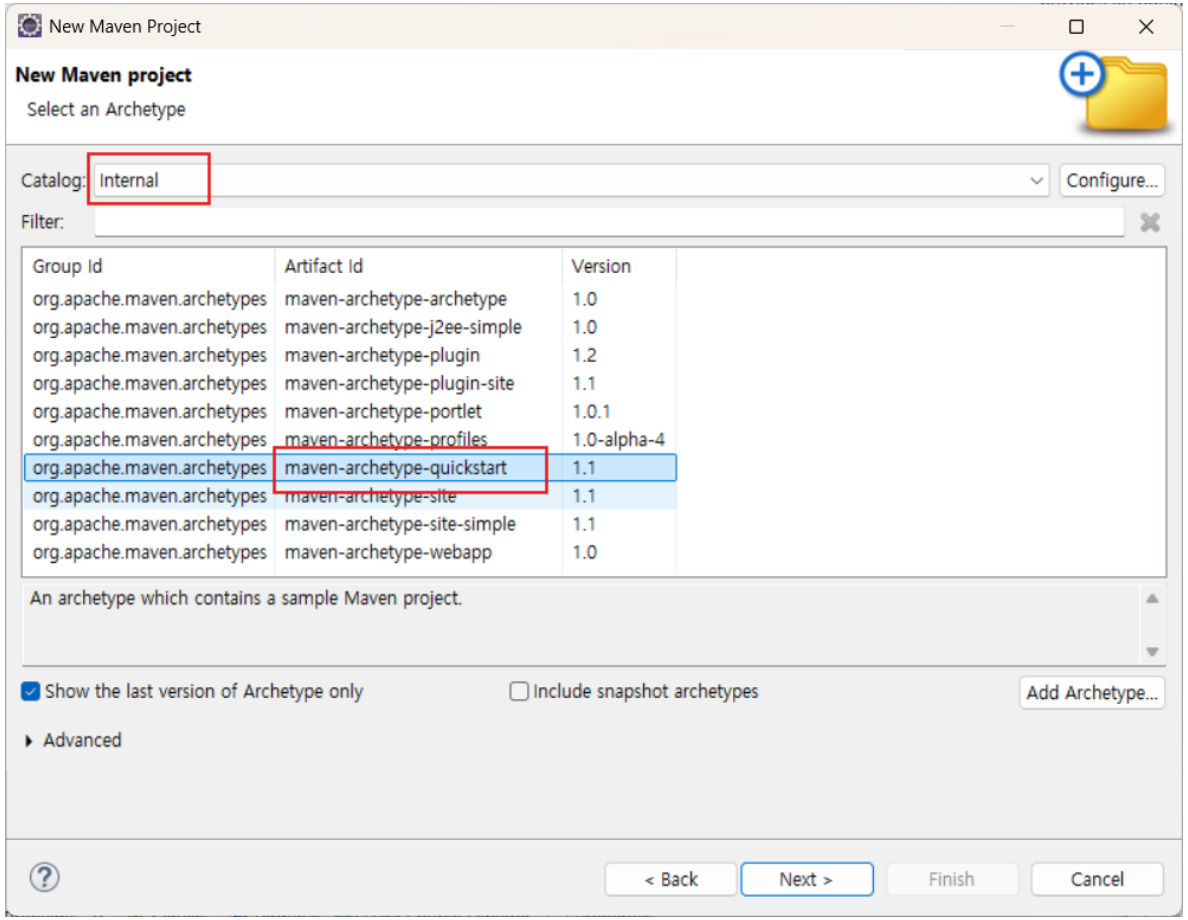
1.1 Mavem Project 생성

- File ➔ New → Maven Project 선택



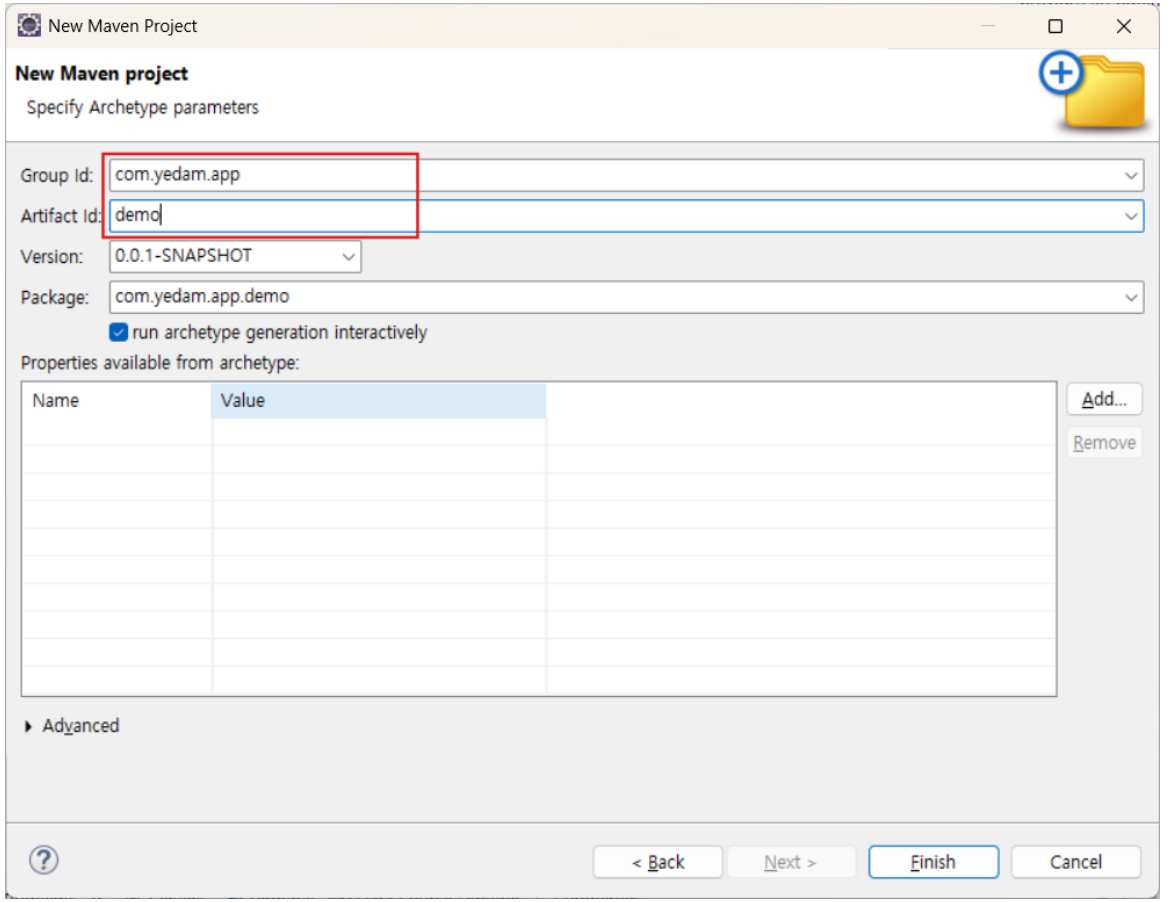
1.1 Mavem Project 생성

- maven-archetype-quickstart 선택



1.1 Mavem Project 생성

- group Id
- Artifact Id
- version
- Package



1.2 pom.xml dependency 설정

- org.springframework-version -> 6.2.7
  - spring-webmvc
  - spring-test
  - spring-aop
  - spring-jdbc
  - spring-tx
- log4j2 -> 2.24.3
- junit -> 4.12
- jakarta.servlet.jsp.jstl-api (jakarta-servlet-api, jakarta-el 포함됨)
- jakarta.servlet.jsp.jstl
- Lombok
- Jackson
- org.aspectj-version -> 1.9.0

1.2 pom.xml dependency 설정

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>com.yedam</groupId>
<artifactId>sp06_mvc</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>war</packaging>
<properties>
  <springframework.version>6.2.7</springframework.version>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>
<dependencies>
  <!-- spring-webmvc -->
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>${springframework.version}</version>
  </dependency>
  <!-- spring-test -->
 <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-test</artifactId>
    <version>${springframework.version}</version>
  </dependency>
  <!-- Lombok -->
  <dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.38</version>
    <scope>provided</scope>
  </dependency>
```

```
<!-- jakarta.servlet-api, jakarta-el 포함됨 -->
  <dependency>
    <groupId>jakarta.servlet.jsp.jstl</groupId>
    <artifactId>jakarta.servlet.jsp.jstl-api</artifactId>
     <version>3.0.0</version>
  </dependency>
  <dependency>
    <groupId>org.glassfish.web</groupId>
    <artifactId>jakarta.servlet.jsp.jstl</artifactId>
    <version>3.0.1</version>
  </dependency>
  <!-- junit -->
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.13.2</version>
    <scope>test</scope>
  </dependency>
  <!-- log4j2 -->
  <dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-slf4j2-impl</artifactId>
    <version>2.24.3</version>
  </dependency>
</dependencies>
</project>
```

1.2 pom.xml dependency 설정

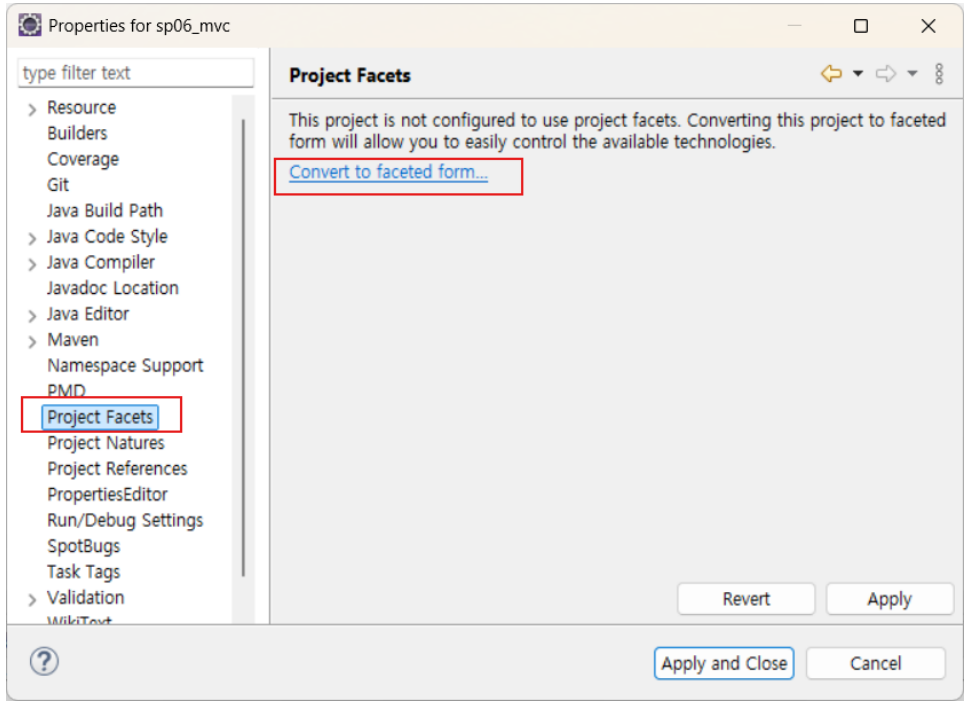
- spring 5 → tomcat 9 → web module 3.1

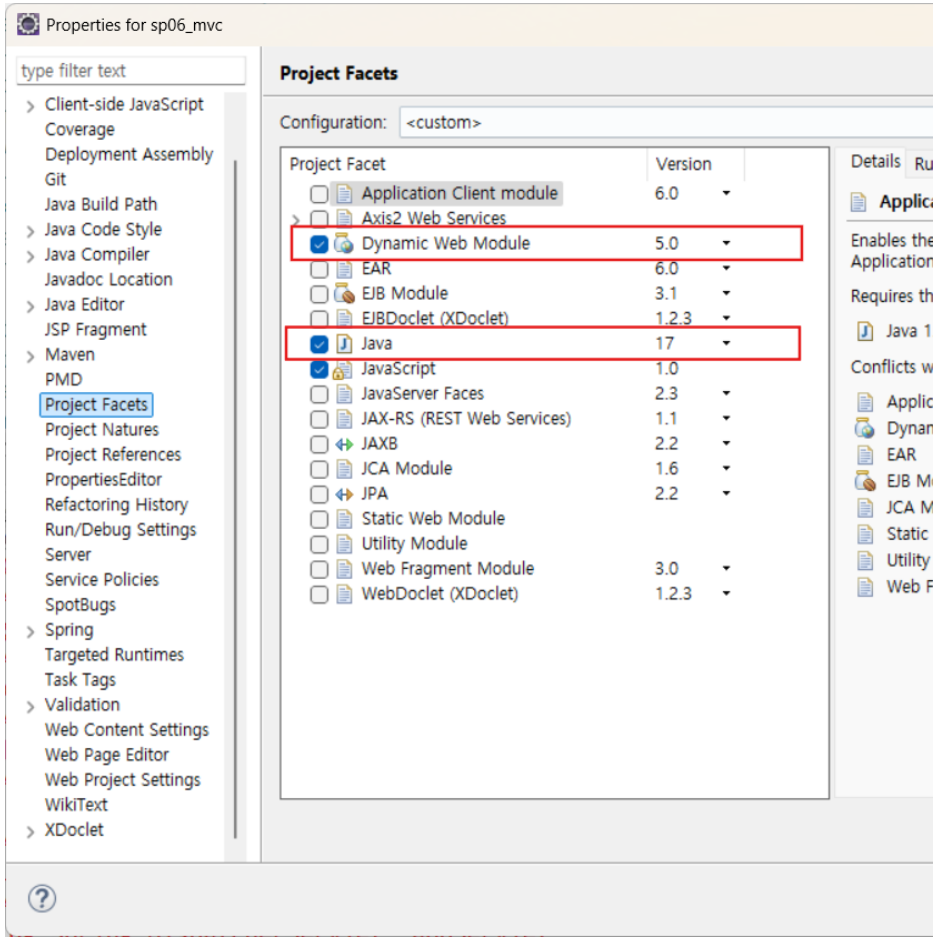
```
<properties>
<springframework.version>5.3.39</springframework.version>
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>
```

```
<!-- Servlet -->
<dependency>
<groupId>javax.servlet</groupId>
<artifactId>javax.servlet-api</artifactId>
<version>3.1.0</version>
<scope>provided</scope>
</dependency>
<dependency>
<groupId>javax.servlet.jsp</groupId>
<artifactId>javax.servlet.jsp-api</artifactId>
<version>2.3.3</version>
<scope>provided</scope>
</dependency>
<dependency>
<groupId>javax.servlet</groupId>
<artifactId>jstl</artifactId>
<version>1.2</version>
</dependency>
```

1.3 project version 변경

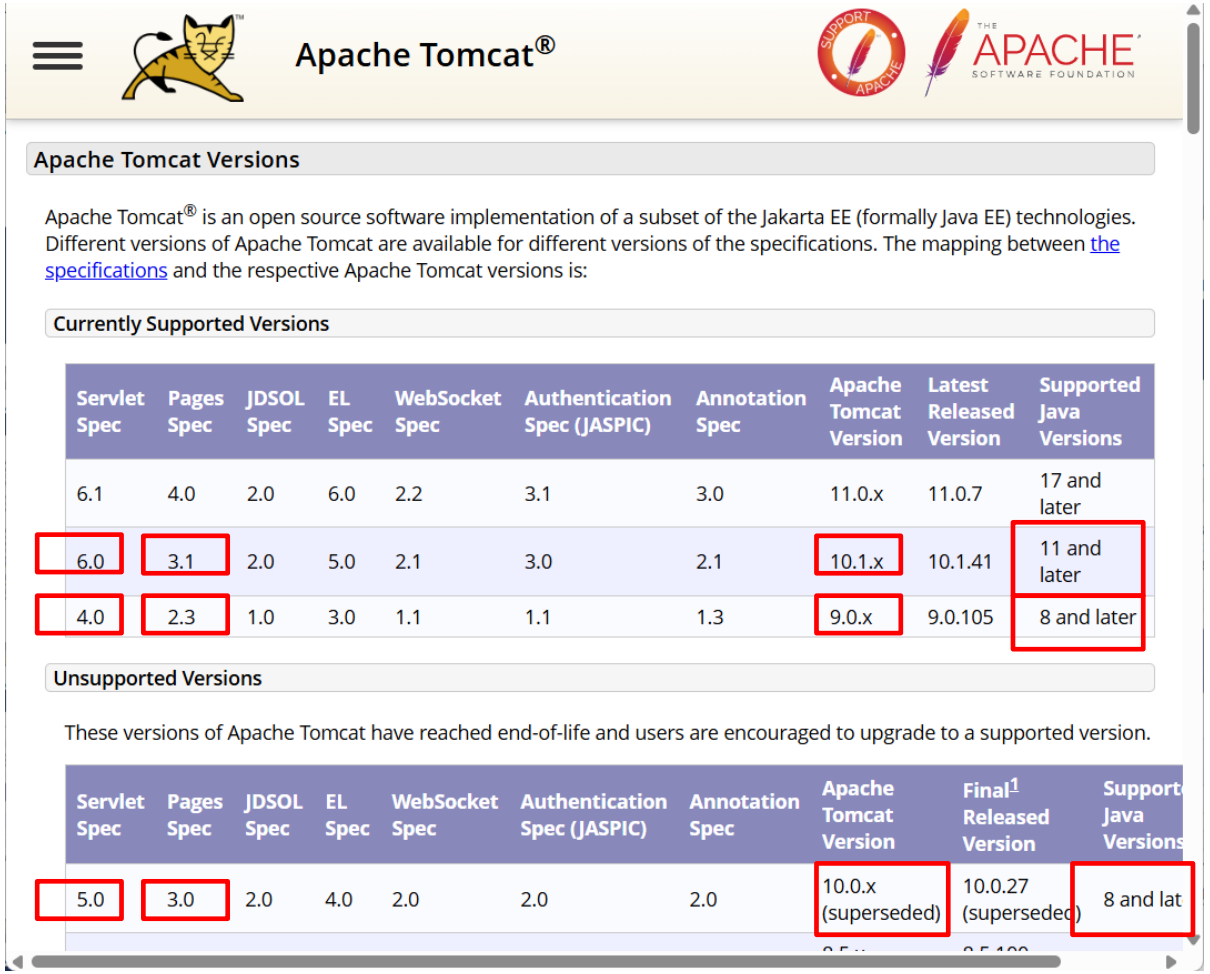
- properties -> Project facets





1.3 톰캣 version 별로 웹모듈 버전 지정

- https://tomcat.apache.org/whichversion.html



Apache Tomcat®

Apache Tomcat Versions

Apache Tomcat® is an open source software implementation of a subset of the Jakarta EE (formally Java EE) technologies. Different versions of Apache Tomcat are available for different versions of the specifications. The mapping between the specifications and the respective Apache Tomcat versions is:

Currently Supported Versions

Servlet Spec	Pages Spec	JDSOL Spec	EL Spec	WebSocket Spec	Authentication Spec (JASPIC)	Annotation Spec	Apache Tomcat Version	Latest Released Version	Supported Java Versions
6.1	4.0	2.0	6.0	2.2	3.1	3.0	11.0.x	11.0.7	17 and later
6.0	3.1	2.0	5.0	2.1	3.0	2.1	10.1.x	10.1.41	11 and later
4.0	2.3	1.0	3.0	1.1	1.1	1.3	9.0.x	9.0.105	8 and later

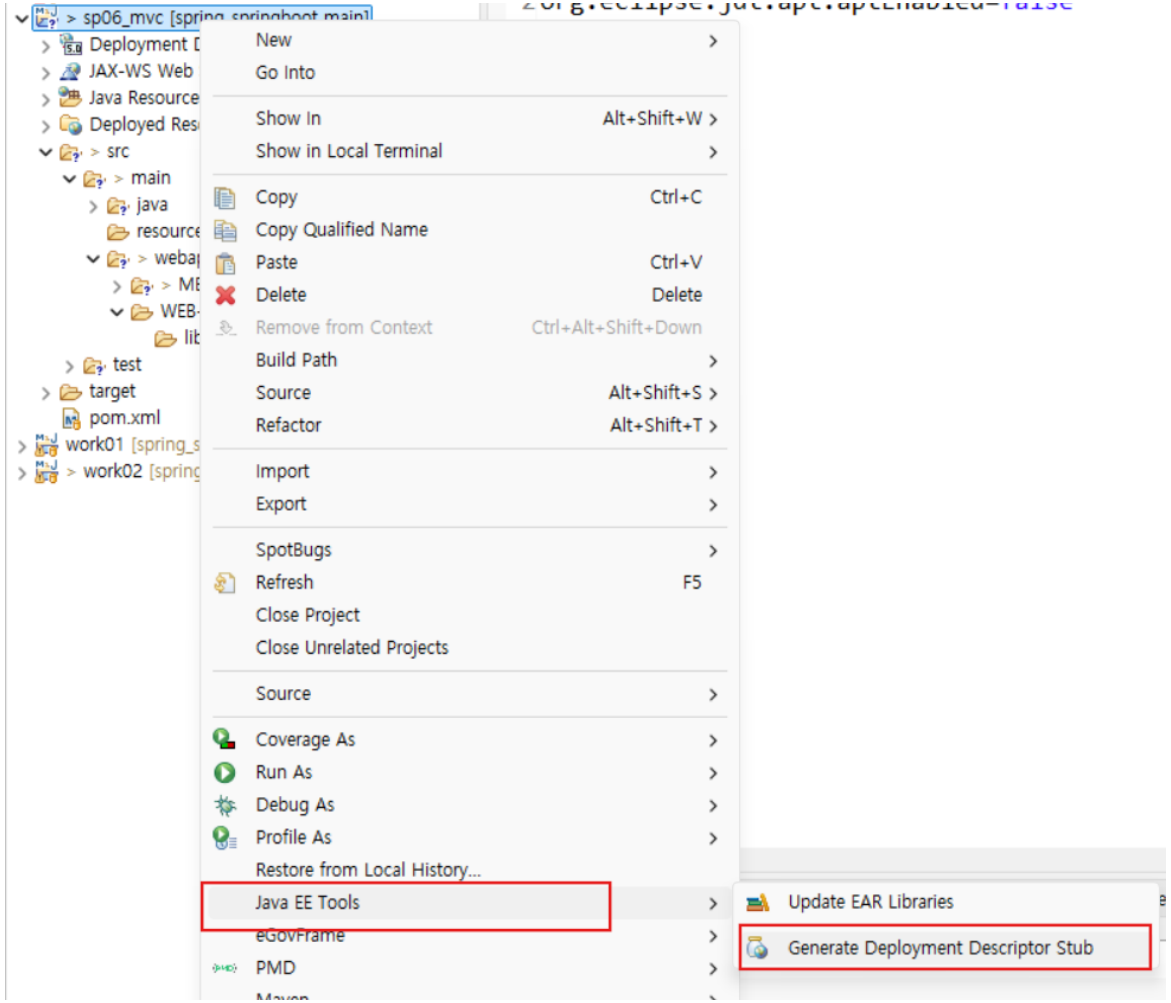
Unsupported Versions

These versions of Apache Tomcat have reached end-of-life and users are encouraged to upgrade to a supported version.

Servlet Spec	Pages Spec	JDSOL Spec	EL Spec	WebSocket Spec	Authentication Spec (JASPIC)	Annotation Spec	Apache Tomcat Version	Final[1] Released Version	Support Java Versions
5.0	3.0	2.0	4.0	2.0	2.0	2.0	10.0.x (superseded)	10.0.27 (superseded)	8 and lat

1.4 web.xml

- web.xml 추가
  - 프로젝트 컨텍스트메뉴 → Java EE Tools → Generate Deployment Descriptor Sub



1.4 web.xml

- \src\main\webapp\WEB-INF\web.xml
  - namespace 톰캣 버전에 맞게 지정

```
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="https://jakarta.ee/xml/ns/jakartaee"
xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee https://jakarta.ee/xml/ns/jakartaee/web-app_5_0.xsd"  version="5.0">
```


```
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd" version="4.0">
```

```
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd" version="3.1">
```

1.4 web.xml

```
<!-- 스프링 디스패처 서블릿 매핑    -->
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>/WEB-INF/spring/*-context.xml</param-value>
</context-param>
<listener>
  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
<servlet>
  <servlet-name>appServlet</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <init-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>appServlet</servlet-name>
  <url-pattern>/</url-pattern>
</servlet-mapping>
```

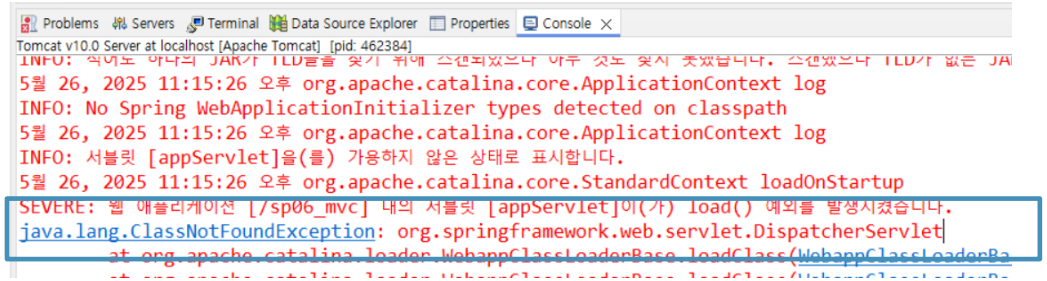
```
<!-- 인코딩 필터 -->
<filter>
  <filter-name>encFilter</filter-name>
  <filter-class>
org.springframework.web.filter.CharacterEncodingF
ilter</filter-class>
  <init-param>
    <param-name>encoding</param-name>
    <param-value>UTF-8</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>encFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

1.5 servlet-context.xml

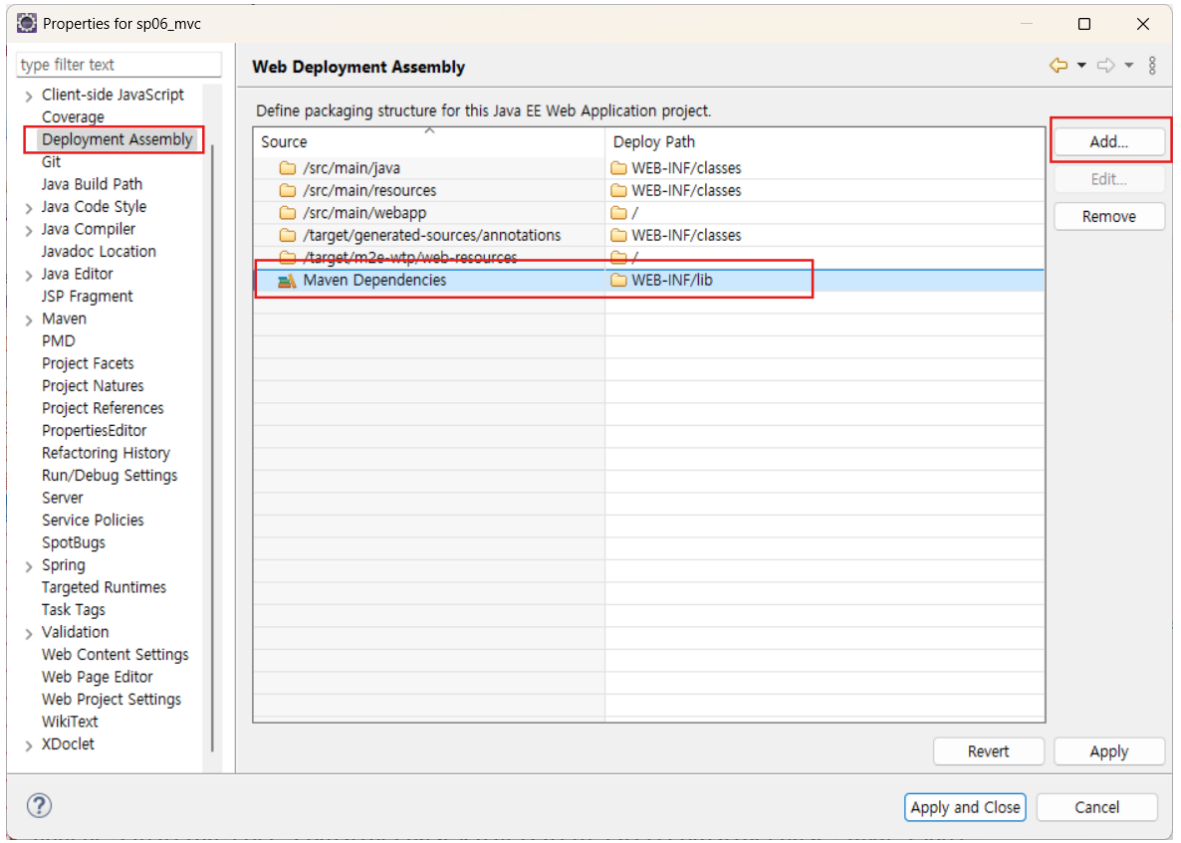
```
<?xml version="1.0" encoding="UTF-8"?>
<beans:beans xmlns="http://www.springframework.org/schema/mvc"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:beans="http://www.springframework.org/schema/beans"
xmlns:context="http://www.springframework.org/schema/context"
xsi:schemaLocation="http://www.springframework.org/schema/mvc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

1.5 Web Deployment Assembly

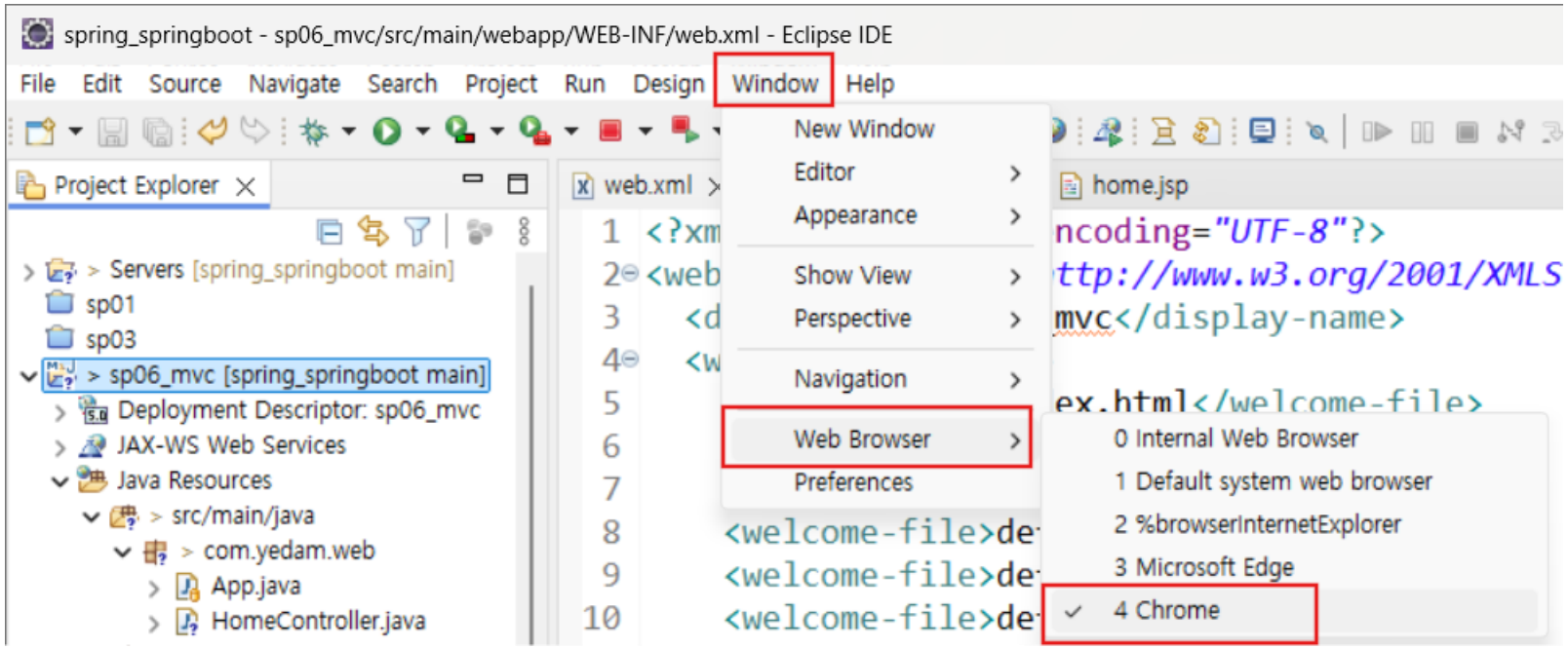
- DispatcherServlet 에러가 나는 경우
  - Deployment Assembly → add 버튼 → Java Build Path Entries 타입선택 → maven ependency 선택



- Deployment Assembly 에 Maven Dependency 추가



1.6 브라우저 지정


spring_springboot - sp06_mvc/src/main/webapp/WEB-INF/web.xml - Eclipse IDE
File Edit Source Navigate Search Project Run Design Window Help
New Window
Editor
Appearance
Show View
Perspective
Navigation
Web Browser
Preferences
0 Internal Web Browser
1 Default system web browser
2 %browserInternetExplorer
3 Microsoft Edge
4 Chrome
Project Explorer
Servers [spring_springboot main]
sp01
sp03
sp06_mvc [spring_springboot main]
Deployment Descriptor: sp06_mvc
JAX-WS Web Services
Java Resources
src/main/java
com.yedam.web
App.java
HomeController.java
web.xml
home.jsp

2.1 커넥션 풀 설정

- pom.xml 에 <dependency> 추가
  - HikariCP
  - spring-jdbc (spring-tx 포함됨)
  - ojdbc8

```
<!-- Database connection pool -->
<dependency>
<groupId>com.zaxxer</groupId>
<artifactId>HikariCP</artifactId>
<version>5.0.1</version>
</dependency>

<!-- spring-jdbc -->
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-jdbc</artifactId>
<version>${org.sprigframework-version}</version>
</dependency>

<!-- ojdbc8 -->
<dependency>
<groupId>com.oracle.database.jdbc</groupId>
<artifactId>ojdbc8</artifactId>
<version>19.3.0.0</version>
</dependency>
```

- src\main\webapp\WEB-INF\spring\root-context.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:mybatis-spring="http://mybatis.org/schema/mybatis-spring"
xsi:schemaLocation="http://mybatis.org/schema/mybatis-spring
http://mybatis.org/schema/mybatis-spring-1.2.xsd
http://www.springframework.org/schema/beans
https://www.springframework.org/schema/beans/spring-beans.xsd">

<!-- datasource connection pool -->
<bean id="hikariConfig" class="com.zaxxer.hikari.HikariConfig">
  <property name="driverClassName" value="oracle.jdbc.driver.OracleDriver" />
  <property name="jdbcUrl"    value="jdbc:oracle:thin:@127.0.0.1:1521:xe" />
  <property name="username"   value="hr" />
  <property name="password"   value="hr" />
</bean>

<bean id="dataSource" class="com.zaxxer.hikari.HikariDataSource"
      destroy-method="close">
  <constructor-arg ref="hikariConfig" />
</bean>

</beans>
```

2.2 Mybatis 설정

- pom.xml 에 <dependency> 추가
  - mybatis-spring
  - mybatis

```
<!-- mybatis -->
<dependency>
<groupId>org.mybatis</groupId>
<artifactId>mybatis</artifactId>
<version>3.5.19</version>
</dependency>

<!-- mybatis-spring -->
<dependency>
<groupId>org.mybatis</groupId>
<artifactId>mybatis-spring</artifactId>
<version>3.0.4</version>
</dependency>
```

- src\main\resources\spring\datasource-context.xml

```
<!-- mybatis  SqlSessionFactory -->
<bean class="org.mybatis.spring.SqlSessionFactoryBean">
  <property name="dataSource" ref="dataSource"/>
</bean>

<!– mapper scan -->
<mybatis-spring:scan base-package="com.company.**.mapper" />
```

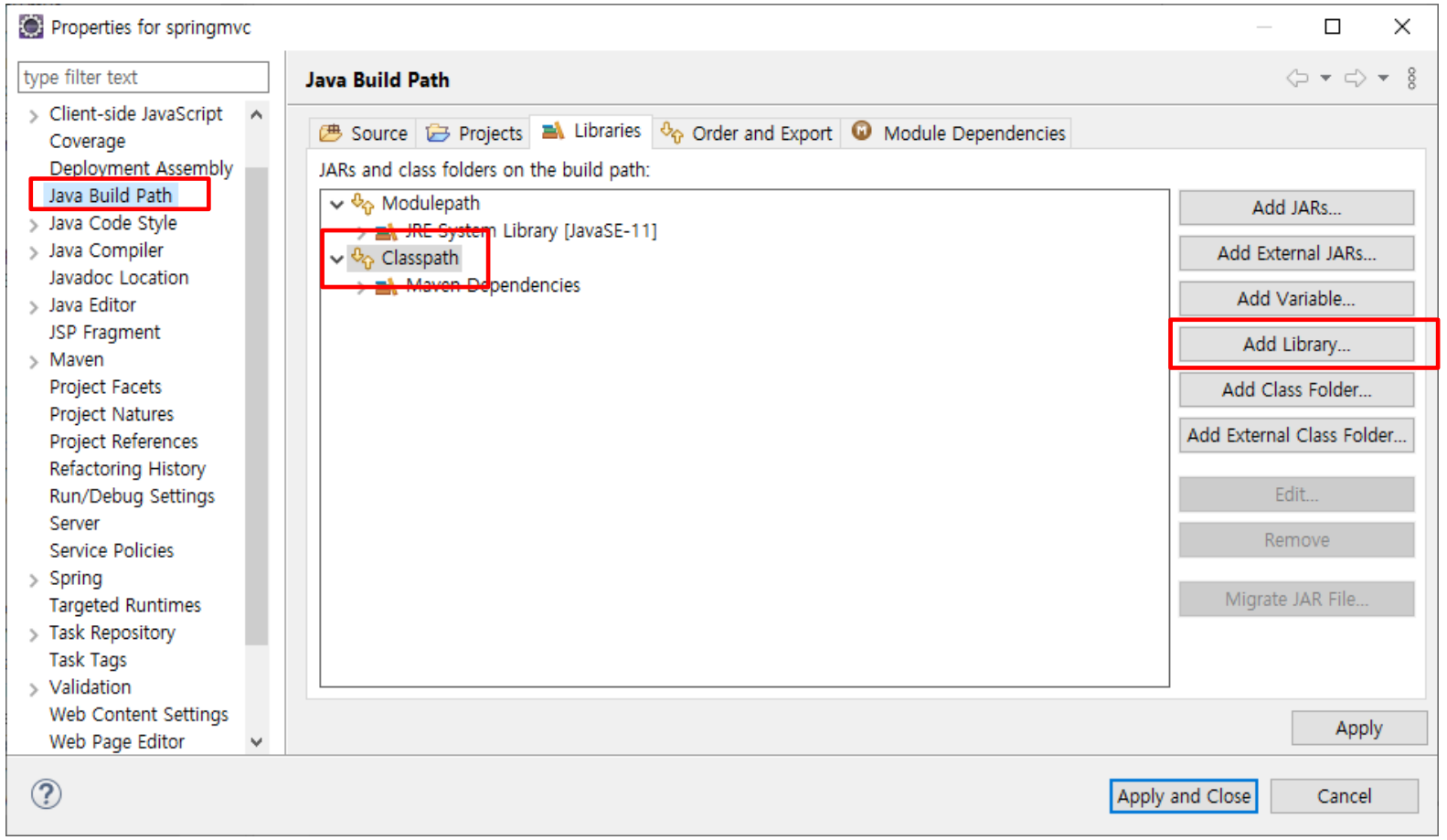
2.1 Mybatis 설정

- src\main\resources\spring\spring\datasource-context.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:mybatis-spring="http://mybatis.org/schema/mybatis-spring"
xsi:schemaLocation="http://mybatis.org/schema/mybatis-spring http://mybatis.org/schema/mybatis-spring-1.2.xsd
http://www.springframework.org/schema/beans https://www.springframework.org/schema/beans/spring-beans.xsd">

<!-- datasource connection pool -->
<bean id="hikariConfig" class="com.zaxxer.hikari.HikariConfig">
  <property name="driverClassName" value="oracle.jdbc.driver.OracleDriver" />
  <property name="jdbcUrl"    value="jdbc:oracle:thin:@127.0.0.1:1521:xe" />
  <property name="username"   value="hr" />
  <property name="password"   value="hr" />
</bean>
<bean id="dataSource" class="com.zaxxer.hikari.HikariDataSource" destroy-method="close">
  <constructor-arg ref="hikariConfig" />
</bean>

<!-- mybatis  SqlSessionFactory -->
<bean class="org.mybatis.spring.SqlSessionFactoryBean">
  <property name="dataSource" ref="dataSource"/>
    <property name="mapperLocations" value="classpath:mapper/*.xml" />
</bean>
<mybatis-spring:scan base-package="com.yedam.**.mapper"></mybatis-spring:scan>
</beans>
```

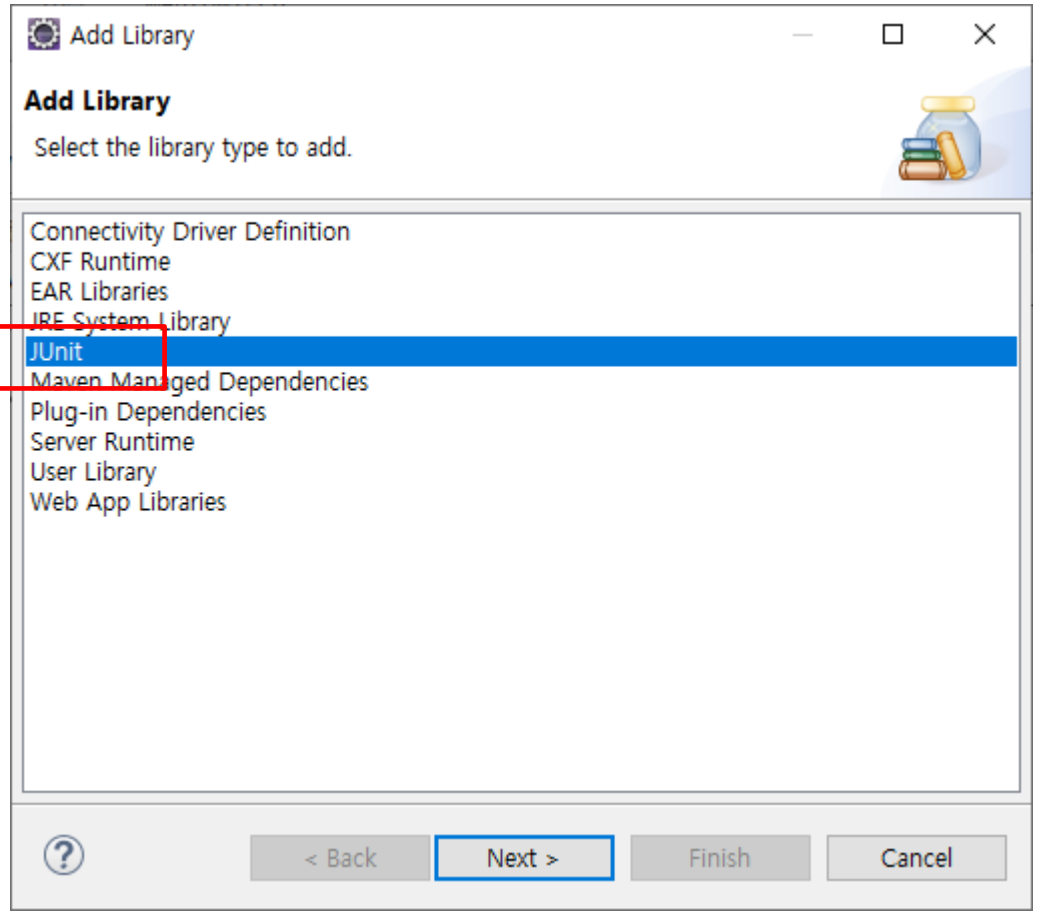
3.1 junit 라이브러리 추가

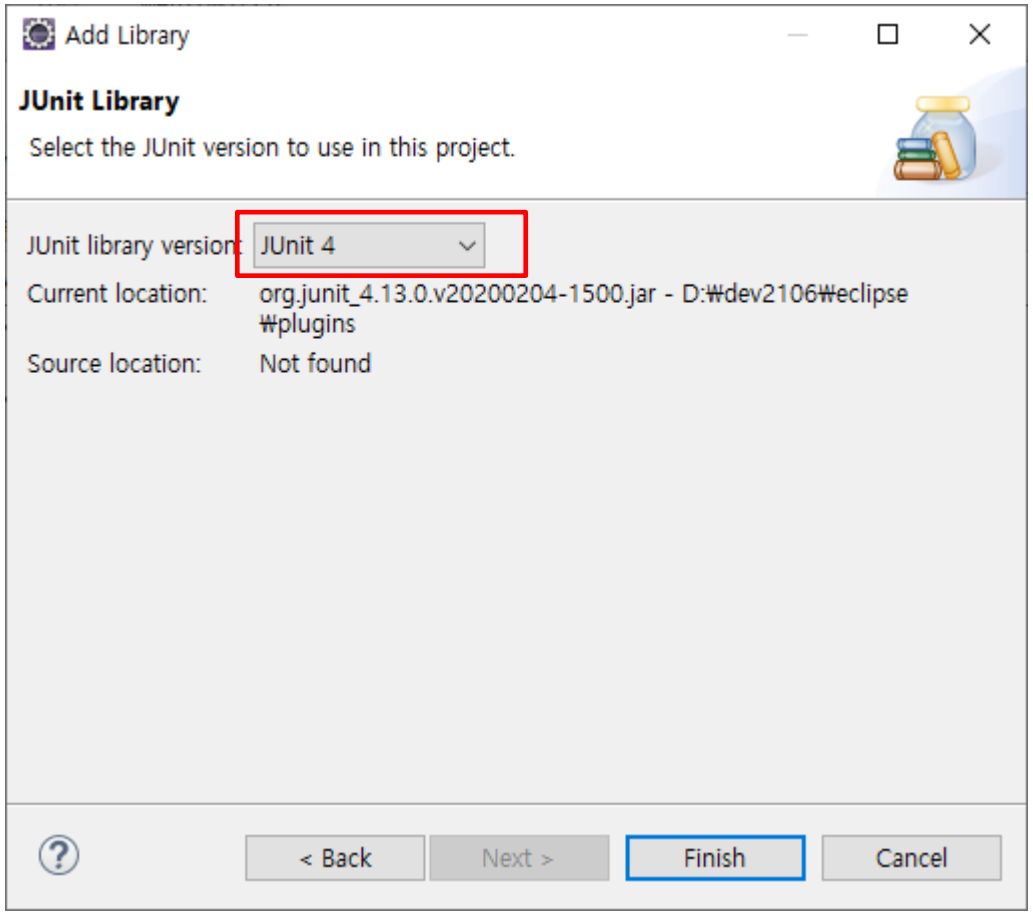
- Properties -> Java BuildPath -> Libraries 탭



3.1 junit 라이브러리 추가

- junit 라이브러리 추가





3.2 spring-test 라이브러리 추가

- Pom.xml

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.13.2</version>
</dependency>


<dependency>
<groupId>org.springframework</groupId>
  <artifactId>spring-test</artifactId>
  <version>${org.springframework-version}</version>
</dependency>
```

3. 3 spring 테스트 어노테이션

- Spring-test에서 테스트를 지원하는 어노테이션
  - @RunWith(SpringJUnit4ClassRunner.class)
    - @WunWith는 jUnit 프레임워크의 테스트 실행방법을 확장할 때 사용하는 어노테이션이다.
    - SpringJUint4ClassRunner라는 클래스를 지정해주면 ApplicationContext를 만들고 관리하는 작업을 진행해준다.
    - @RunWith 어노테이션은 각가의 테스트별로 객체가 생성되더라도 싱글톤(singleton)의 Application Context를 보장한다.
  - @ContextConfiguration
    - 스프링 빈(Bean) 설정 파일의 위치를 지정할 때 사용되는 어노테이션이다.

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations = "classpath:spring/*-context.xml")
public class BoardClient {
  @Autowired ApplicationContext context;

  @Test
  public void dataSourceTest() throws SQLException {
      DataSource ds = (DataSource) context.getBean("dataSource");
      System.out.println(ds.getConnection());
  }
```

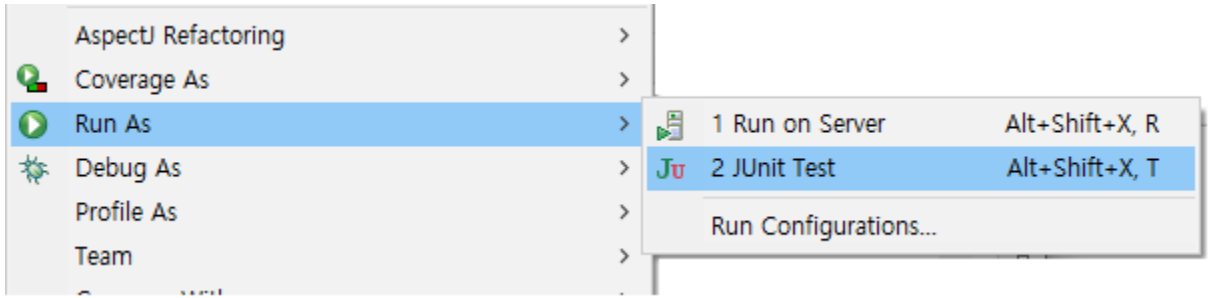
3.4 테스트 코드

- src/test/java/EmpMapperTest.java

```
import static org.junit.Assert.assertEquals;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations = "file:src/main/java/resources/spring/datasource-context.xml")
public class EmpMapperClient {

    @Autowired EmpMapper empMapper;

    @Test
    public void getEmp() {
        EmpVO vo = new EmpVO();
        vo.setEmployee_id("100");
        EmpVO findVO = empMapper.getEmp(vo);
        assertEquals(findVO.getLast_name(),"King");
    }
}
```

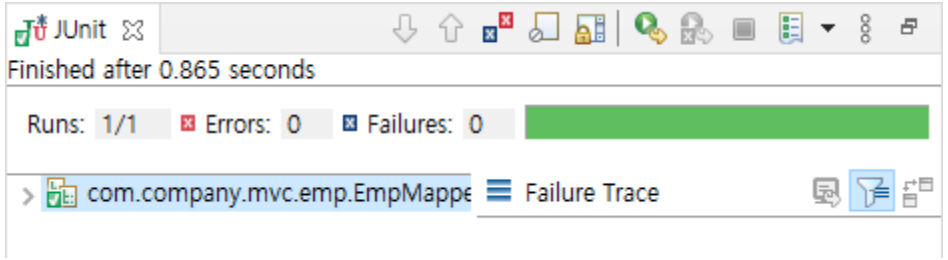
3.5 junit 테스트

- jUnit Test

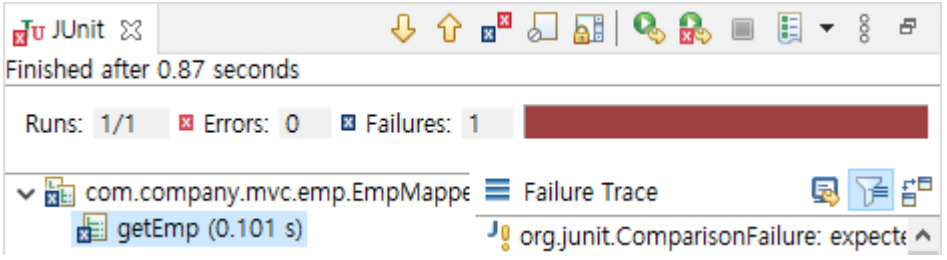


- 실행결과

테스트 성공



테스트 실패



3.6 jUnit 개요와 특징

- jUnit의 개요
  - Java에서 독립된 단위테스트(unit Test)를 지원해주는 프레임워크
  - 단위테스트란 소스 코드의 특정 모듈이 의도된 대로 정확히 작동하는지 검증하는 절차, 즉 모든 함수와 메소드에 대한 테스트 케이스(Test case)를 작성하는 절차

- jUnit의 특징
  - TDD의 창시자인 Kent Beck과 디자인 패턴 책의 저자인 Eric Gamma가 작성
  - 단정(assert) 메서드로 테스트 케이스의 수행결과를 판별한다.
    - 예) assertEquals(예상값, 실제값)
  - jUnit4부터는 테스트를 지원하는 어노테이션을 제공한다.
    - @Test, @Before, @After
  - 각 @Test 메서드가 호출될 때마다 새로운 인스턴스를 생성하여 독립적인 테스트가 이루어지도록 한다.
  - 결과는 성공(녹색), 실패(붉은색)중 하나로 표시

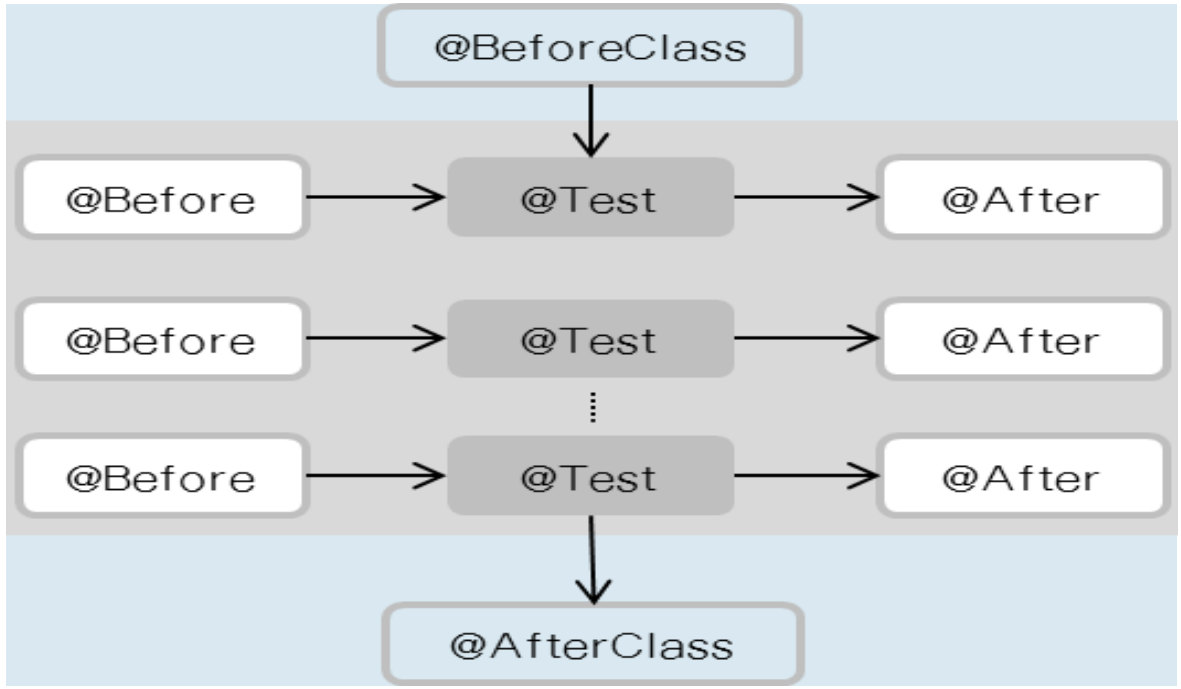
3.7 jUnit에서 테스트를 지원하는 어노테이션

- @Test
  - @Test가 선언된 메소드는 테스트를 수행하는 메소드가 된다.
  - Junit은 각각의 테스트가 서로 영향을 주지 않고 독립적으로 실행됨을 원칙으로 하므로 @Test마다 객체를 생성한다.
- @Ignore
  - @Ignore가 선언된 메소드는 테스트를 실행하지 않게 한다.
- @Before
  - Before가 선언된 메소드는 @Test 메소드가 실행되지 전에 먼저 실행
  - @Test 메소드가 공통으로 사용하는 코드를 @Before 메소드에 선언하여 사용

3.8 jUnit에서 테스트를 지원하는 어노테이션

@Test 메소드보다 먼저 한번만 수행되어야 할 경우에 사용

@Test 메소드를 실행하기 전에 실행



@Test 메소드가 실행된 후 실행

@Test 메소드보다 나중에 한번만 수행되어야 할 경우에 사용

3.8 jUnit을 사용한 테스트

- 테스트 결과를 확인하는 단정(assert) 메서드
  - assertArrayEquals(a,b)
    - 배열 a와b가 일치함을 확인
  - assertEquals(a,b)
    - 객체 a와b의 값이 같은지 확인
  - assertSame(a,b)
    - 객체 a와b가 같은 객체임을 확인
  - assertTrue(a)
    - a가 참인지 확인
  - assertNotNull(a)
    - a객체가 null이 아님을 확인

http://junit.sourceforge.net/javadoc/org/junit/Assert.html

4.1 로그 라이브러리 설치

pom.xml

```
<!-- SLF4J → Log4j Bridge (Spring은 SLF4J를 통해 로깅) -->
<!-- log4j-core, log4j-api 같이 설치됨 -->
<dependency>
  <groupId>org.apache.logging.log4j</groupId>
  <artifactId>log4j-slf4j2-impl</artifactId>
  <version>2.23.1</version>
</dependency>
```

src/main/resources/log4j2.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration status="WARN">
<Appenders>
  <Console name="Console" target="SYSTEM_OUT">
    <PatternLayout pattern="%d{HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n"/>
  </Console>
</Appenders>
<Loggers>
  <Root level="info">
    <AppenderRef ref="Console"/>
  </Root>
</Loggers>
</Configuration>
```

4.2 로그 설정

- src/main/resources/log4j2.xml
  - 로그 레벨
    - trace < debug < info < warn < error < fetal < off
    - 지정된 레벨 이하는 출력 안됨

- trace : 아주 자세한 로그 출력
- debug : DEV(개발) 환경에서 사용
- error : production(운영서버) 환경
- fatal : 심각한 오류 로그 출력
- off : 로그 출력 안함

  - 루트 로그 레벨 설정
    - 패키지별 별도 지정이 없으면 루트 레벨을 적용함

```
<!-- Root Logger -->
<root>
  <priority value="info" />
  <appender-ref ref="console" />
</root>
```

4.3 로그 출력

LoggerFactory

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class LogTest {
    private static final Logger logger = LoggerFactory.getLogger(LogTest.class);
}
```

Lombok 애노테이션 이용

```
import lombok.extern.log4j.Log4j2;

@Log4j2
public class LogTest {       }
```

4.4 로깅 라이브러리

- 로깅(logging)이란 시스템 동작시 시스템의 상태와 작동 정보를 시간의 경과에 따라 기록하는 것

- System.out.println()을 사용하지 말고 log.XXX() 이용하면 좋은 점
  1. 로그 출력위치(클래스, 메서드, 라인번호)를 알수 있음
  2. 출력 형식을 지정할 수 있음
  3. 성능면에서 훨씬 뛰어남

- 등장 순서

4.4 로깅 라이브러리

- slf4j (Simple Logging Facade For Java)
  - slf4j는 자바 로깅 시스템을 쉽게 사용할 수 있도록 해주는 라이브러리이며, 다양한 자바 로깅 시스템을 사용할 수 있도록 파사드 패턴(facade pattern) 및 추상화를 통해 로깅 기능을 제공
  - 기존에 사용하는 로깅 시스템을 교체하고 싶을 때, 소스 코드를 수정하지 않고도 maven이나 gradle의 의존성 설정만 바꾸면 손쉽게 적용

```
log4j2 구현체 사용 JCL(Jakarta Common Logging 인터페이스)
    ==> jcl-over-slf4j(어댑터)
        ==> SLF4J(인터페이스)
            ==> slf4j-log4j(브릿지)
                ==> log4j2 (slf4j 구현체)
```

- 파사드 패턴(Facade pattern)
  - 복잡한 서브 시스템을 단순한 인터페이스를 제공하여 제어하는 방식. 클라이언트는 복잡한 서브 시스템에 대해 알 필요가 없으며, Facade를 통해 일괄적으로 제어가능하다. 단순한고 일관된 통합 인터페이스르 제공

5.1 컨트롤러와 웹페이지 작성

- Controller

```
@Controller
public class EmpController {

@RequestMapping(value = "/", method = RequestMethod.GET)
public String home(Locale locale, Model model) {
  logger.info("Welcome home! The client locale is {}.", locale);
  Date date = new Date();
  DateFormat dateFormat = DateFormat
        .getDateTimeInstance(DateFormat.LONG,   DateFormat.LONG, locale);
  String formattedDate = dateFormat.format(date);
  model.addAttribute("serverTime", formattedDate );
  return "/home";
}
}
```

- 뷰페이지
  - src/main/webapp/WEB-INF/views/home.jsp

```
<%@ page language="java" contentType="text/html;
charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core"
prefix="c" %>
<html>
<head>
  <title>Home</title>
</head>
<body>
  <h1>Hello world!</h1>
  <a href="getBoardList">게시판</a><br>
  <a href="getUserList">사용자</a><br>
  <P> The time on the server is ${serverTime}. </P>
  <img src="./resources/images/Desert.jpg" width="100"/>
</body>
</html>
```

- tomcat 서버 시작하고 브라우저에서 URL 입력

```
http://localhost/web/emp?employee_id=100
```

6.1 Dynamic Web Project 를 Spring 프로젝트로 변경

- dynamic web project 생성
- Maven 프로젝트로 변경
  - Configure 컨텍스트 메뉴 -> convert to maven project
- Spring 라이브러리 설치
  - https://mvnrepository.com/ 에서 Spring context 검색
  - 5.3.16 버전 선택하여 pom.xml 에 복사
  - Maven Dependencies에서 jar 파일이 추가되었는지 확인
- Spring 설정파일 추가
  - servlet-context.xml
  - root-context.xml
- web.xml 에 DispatcherServlet 매핑 추가