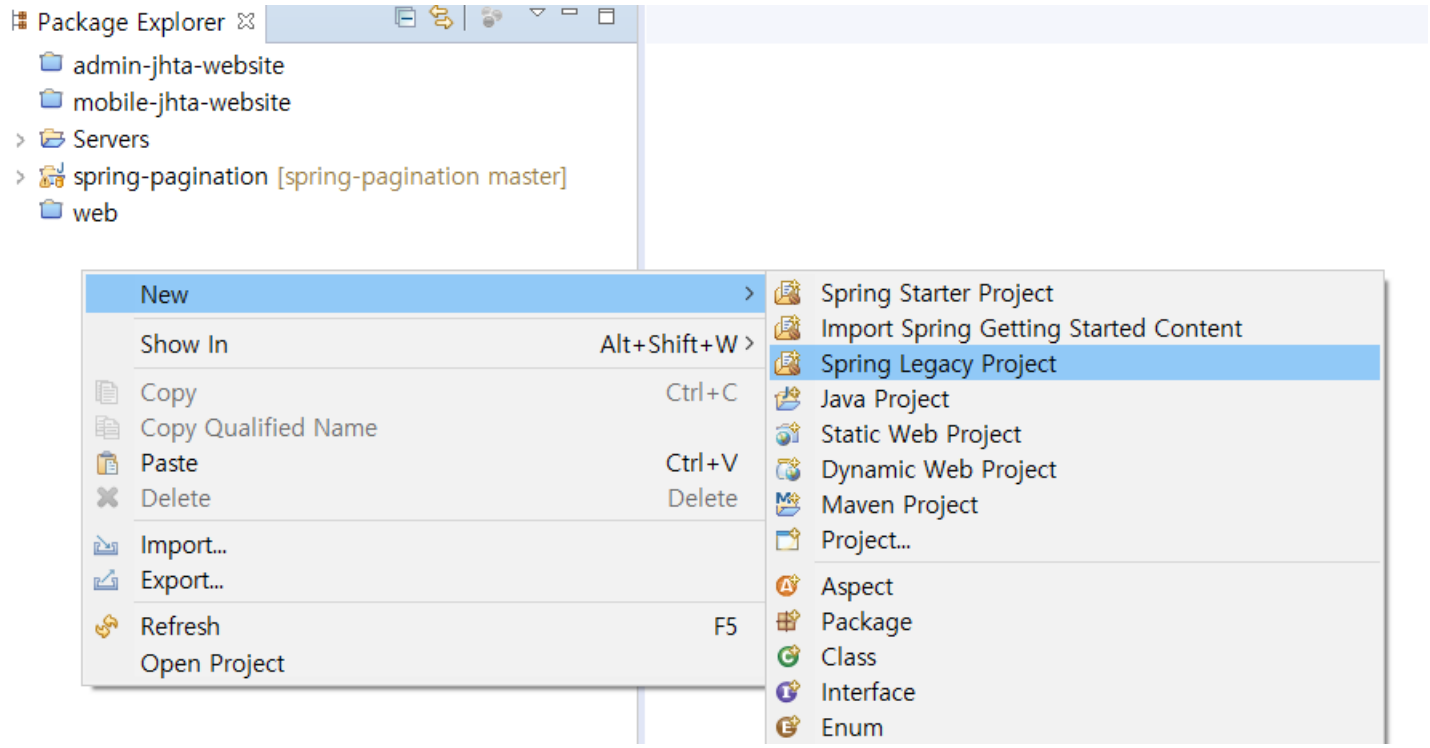
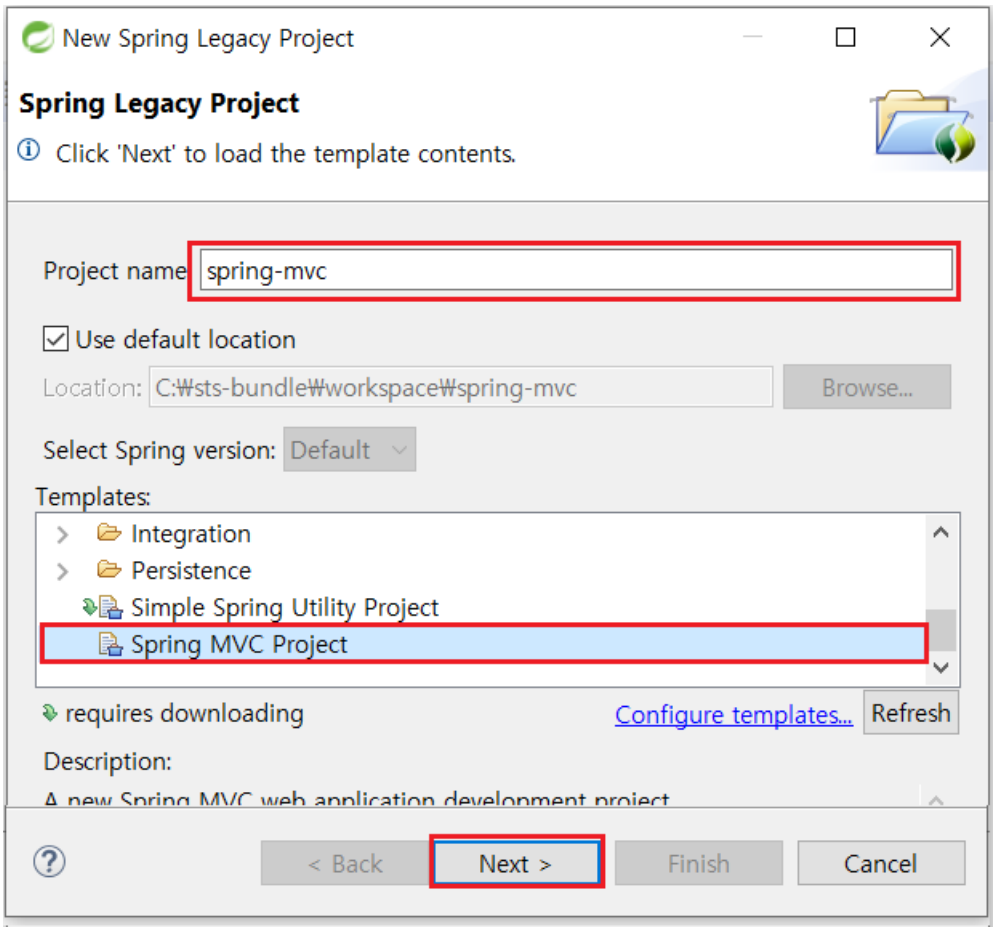
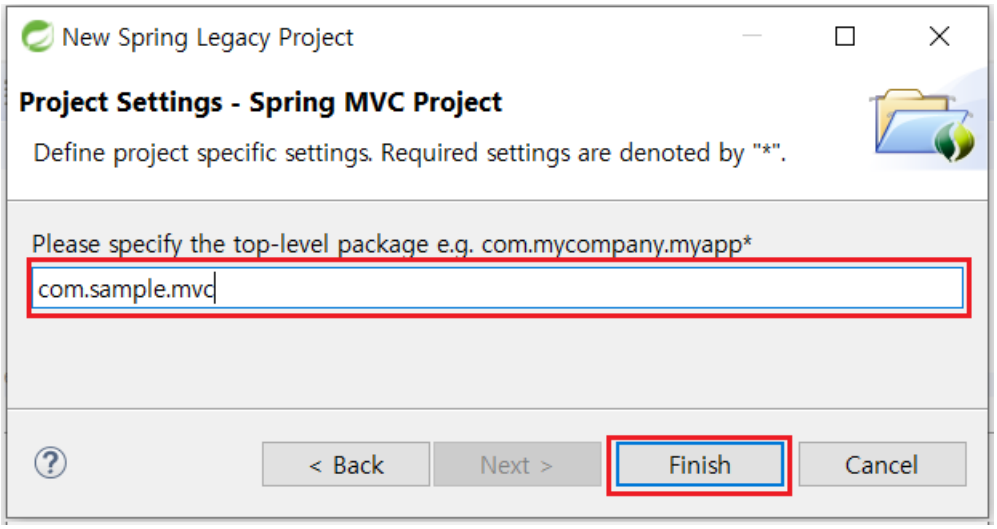
프로젝트 생성하기



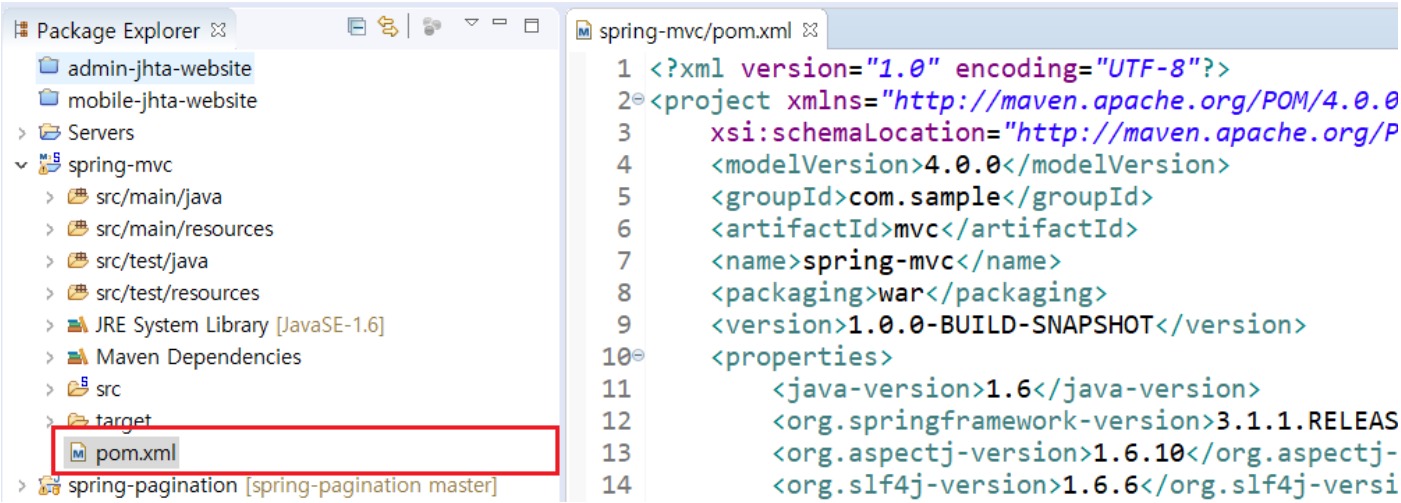
Spring MVC 프로젝트 선택하기



프로젝트의 기본 패키지 정의



프로젝트 설정파일 변경하기



프로젝트 설정파일의 <properties> 설정하기

```
<properties>
    <!--  자바 소스 및 컴파일러 버전 설정하기 -->
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
    <!--  프로젝트 소스 및 출력 인코딩 방식 설정하기 -->
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
    <!--  스프링프레임워크 버전 설정하기 -->
    <springframework-version>4.3.24.RELEASE</springframework-version>
    <!--  Aspectj 버전 설정하기 -->
    <aspectj-version>1.9.4</aspectj-version>
</properties>
```

프로젝트 설정파일의 <repositories> 설정하기

```xml
<repositories>
    <!-- 스프링 JDBC 드라이버를 제공하는 메이븐 원격 저장소 설정하기 -->
    <repository>
        <id>codelds</id>
        <url>https://code.lds.org/nexus/content/groups/main-repo</url>
    </repository>
</repositories>
```

프로젝트 설정파일의 <dependencies>에 스프링프레임워크 관련 의존성 설정하기

```xml
<dependencies>

    <!-- 프로젝트를 위한 스프링프레임워크 의존성 추가하기 -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>${springframework-version}</version>
    </dependency>
    <!-- 프로젝트를 위한 스프링프레임워크 데이터베이스 엑세스 관련 의존성 추가하기-->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-jdbc</artifactId>
        <version>${springframework-version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-tx</artifactId>
        <version>${springframework-version}</version>
    </dependency>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-orm</artifactId>
        <version>${springframework-version}</version>
    </dependency>
    <!-- 프로젝트를 위한 스프링프레임워크 MVC 의존성 추가하기 -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>${springframework-version}</version>
    </dependency>

</dependencies>
```

프로젝트 설정파일의 <dependencies>에 데이터베이스 엑세스 관련 의존성 설정하기

```xml
<!-- 프로젝트를 위한 Oracle JDBC 드라이브 의존성 추가하기 -->
<dependency>
    <groupId>com.oracle</groupId>
    <artifactId>ojdbc6</artifactId>
    <version>11.2.0.3</version>
</dependency>
<!-- 프로젝트를 위한 Connection Pool 관련 의존성 추가하기 -->
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-dbcp2</artifactId>
    <version>2.5.0</version>
</dependency>
<!-- 프로젝트를 위한 mybatis오 mybatis-spring 연동 관련 의존성 추가하기 -->
<dependency>
    <groupId>org.mybatis</groupId>
    <artifactId>mybatis</artifactId>
    <version>3.5.2</version>
</dependency>
<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis-spring</artifactId>
  <version>2.0.2</version>
</dependency>
```

프로젝트 설정파일의 <dependencies>에 기타 의존성 설정하기

```xml
<!-- 프로젝트를 위한 선언적트랜잭션처리에 필요한 AOP관련 의존성 추가하기 -->
<dependency>
    <groupId>org.aspectj</groupId>
    <artifactId>aspectjrt</artifactId>
    <version>${aspectj-version}</version>
</dependency>
<!-- 프로젝트를 위한 파일업로드 지원 의존성 추가하기 -->
<dependency>
    <groupId>commons-fileupload</groupId>
    <artifactId>commons-fileupload</artifactId>
    <version>1.3.3</version>
</dependency>
<!-- 프로젝트를 위한 자바객체와 JSON 텍스트 사이의 변환의 지원하는 의존성 추가하기 -->
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.9.9.1</version>
</dependency>
```

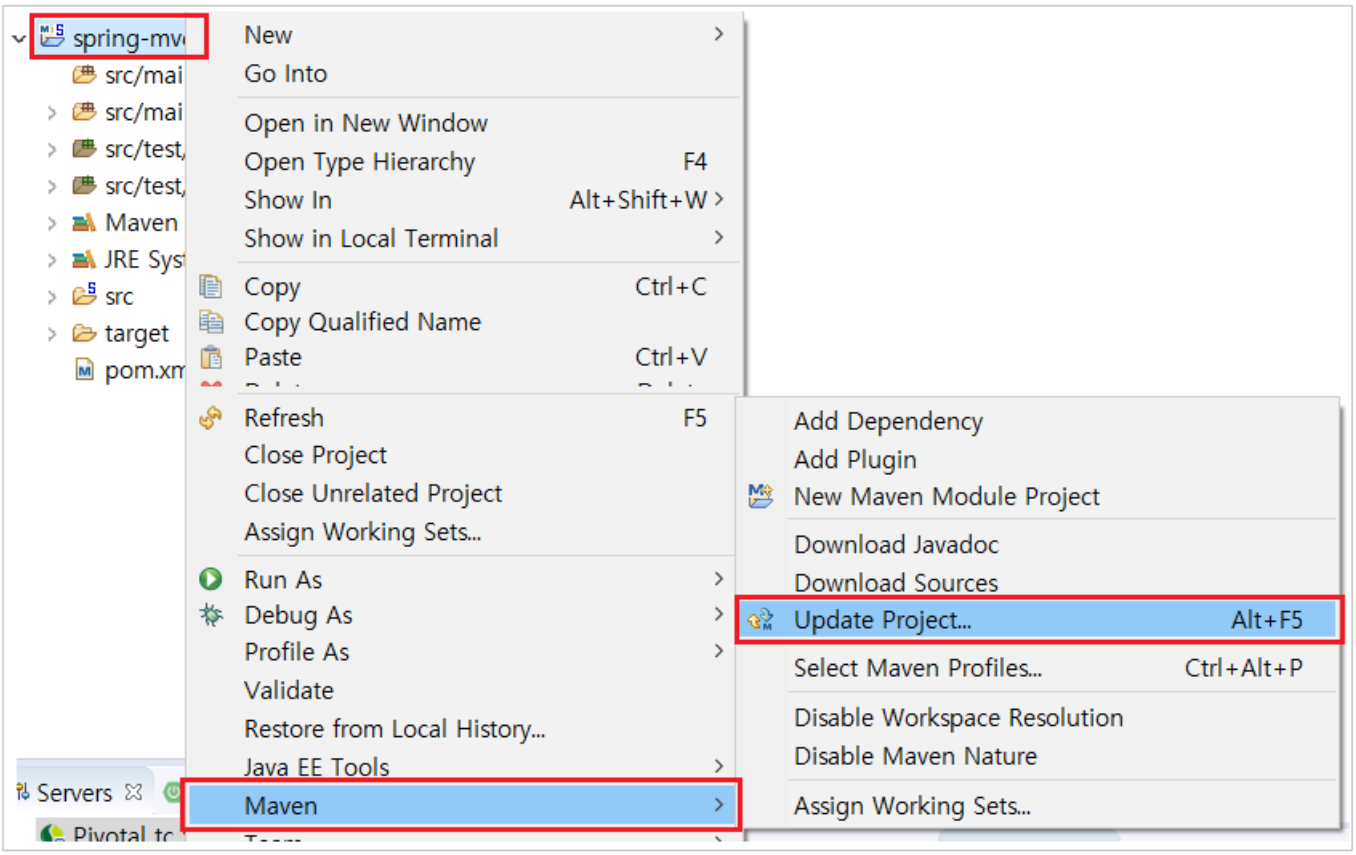
프로젝트 설정파일의 <dependencies>에 로깅관련 의존성 설정하기

```xml
<!-- 프로젝트를 위한 로그출력을 지원하는 의존성 추가하기 -->
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-api</artifactId>
    <version>1.7.28</version>
</dependency>
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>jcl-over-slf4j</artifactId>
    <version>1.7.28</version>
    <scope>runtime</scope>
</dependency>
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-log4j12</artifactId>
    <version>1.7.28</version>
    <scope>runtime</scope>
</dependency>
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-core</artifactId>
    <version>2.12.1</version>
</dependency>
```

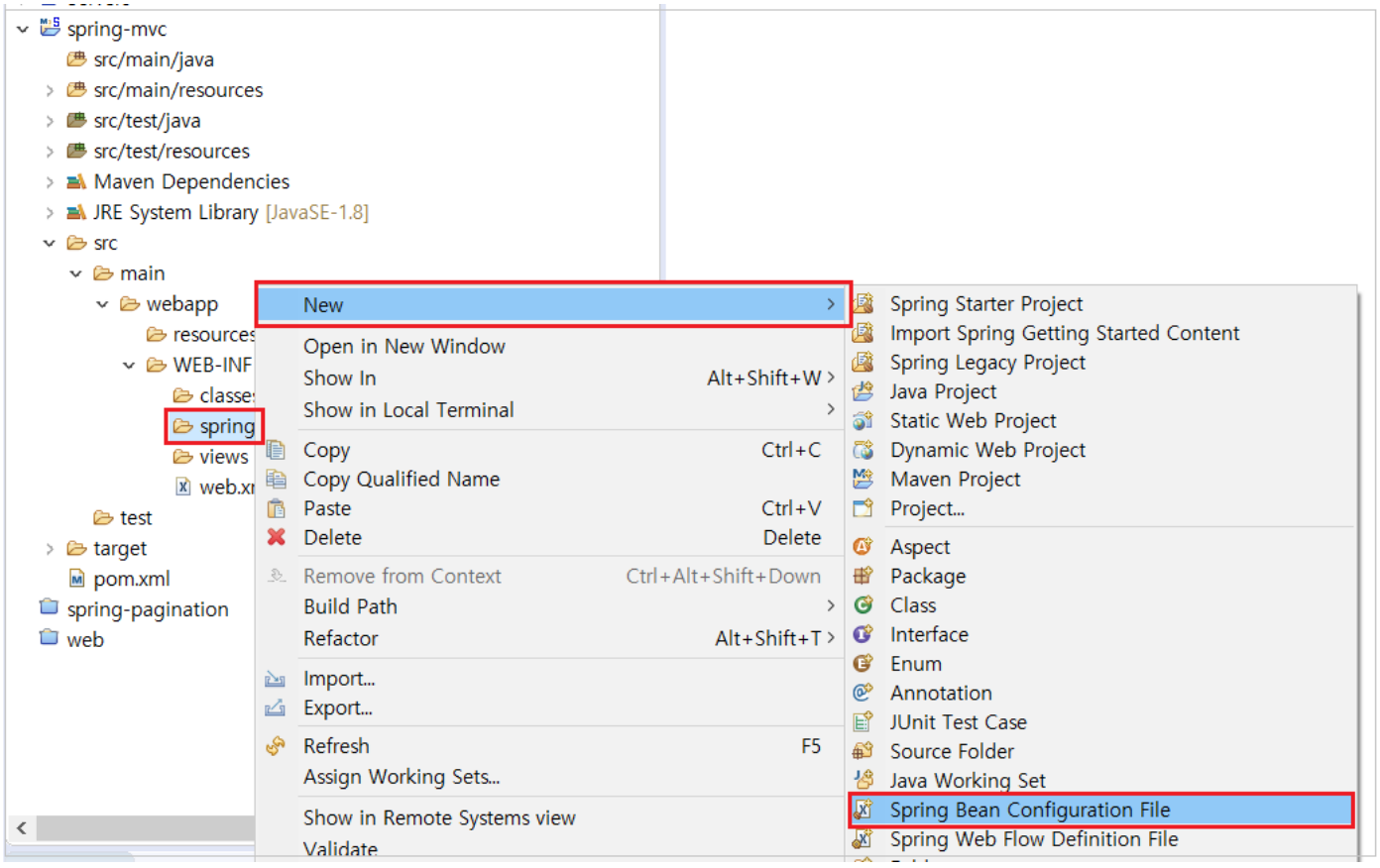
프로젝트 설정파일의 <dependencies>에 웹 관련(서블릿, JSP, JSTL) 의존성 설정하기

```xml
<!-- 프로젝트를 위한 서블릿/JSP/JSTL 의존성 추가하기 -->
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>servlet-api</artifactId>
    <version>2.5</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>javax.servlet.jsp</groupId>
    <artifactId>jsp-api</artifactId>
    <version>2.1</version>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
</dependency>
```

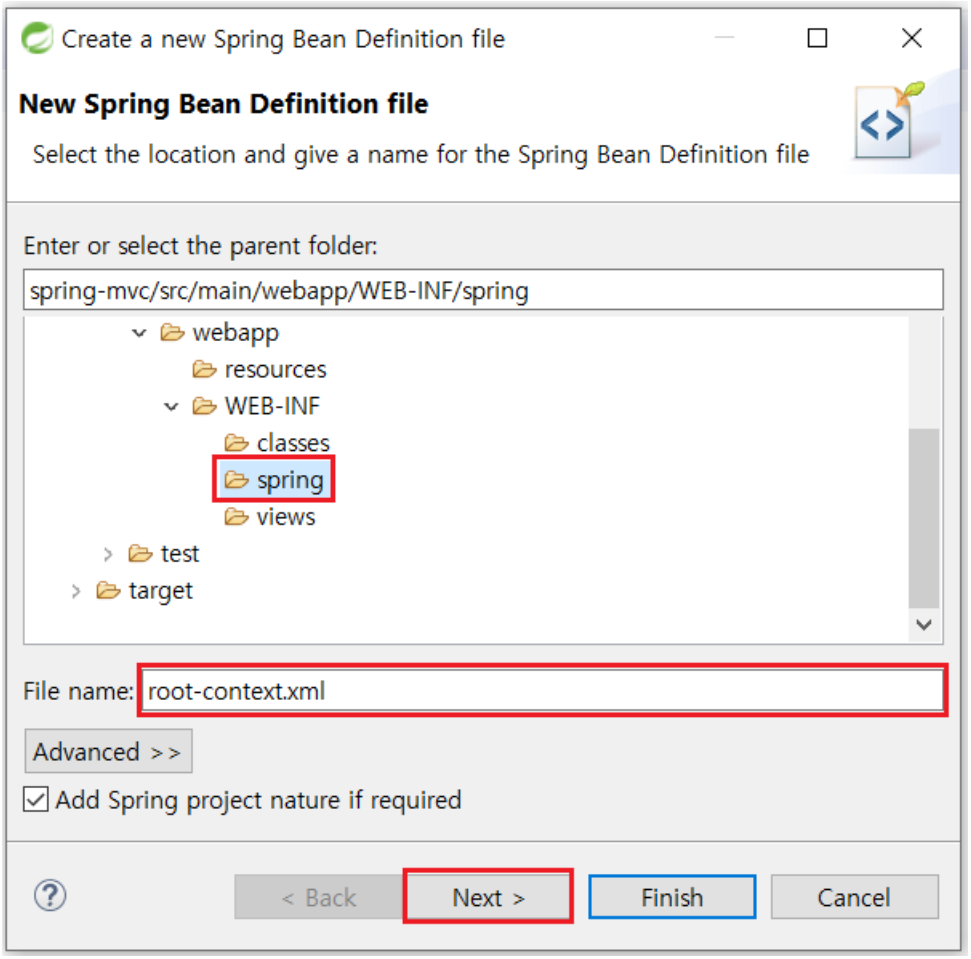
프로젝트 설정파일의 정보를 프로젝트에 반영하기



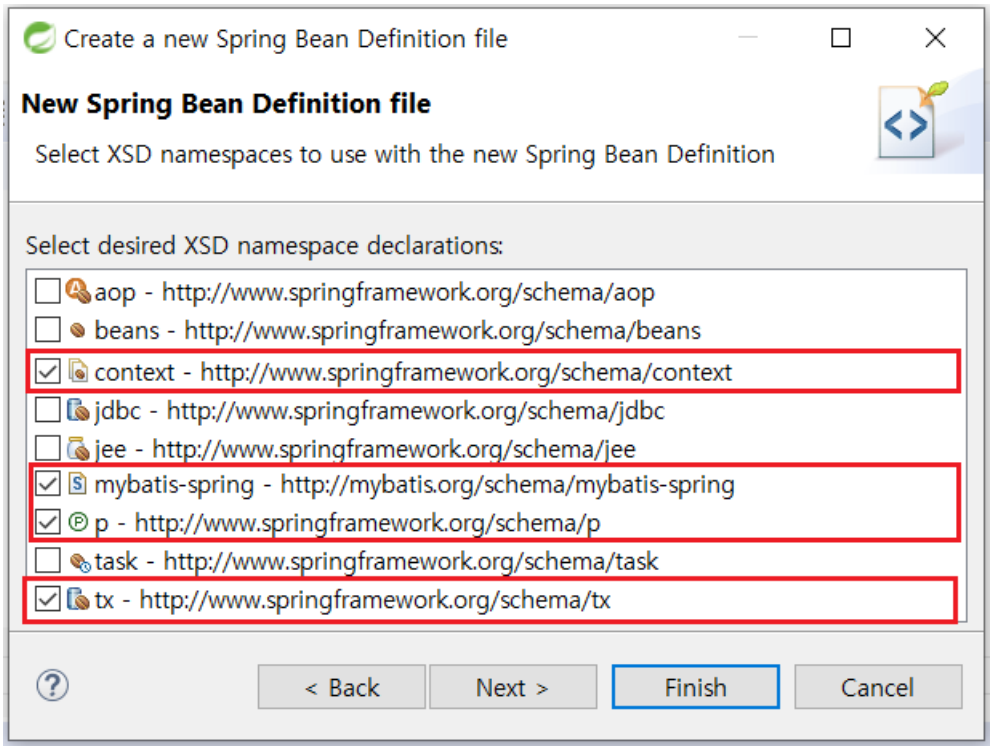
스프링 빈 설정파일(Root 스프링 컨테이너) 생성하기



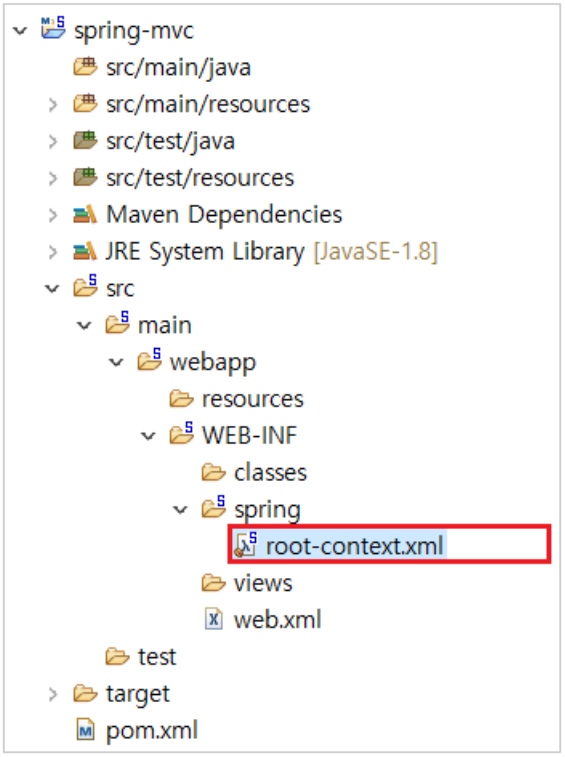
스프링 빈 설정파일 이름 지정하기



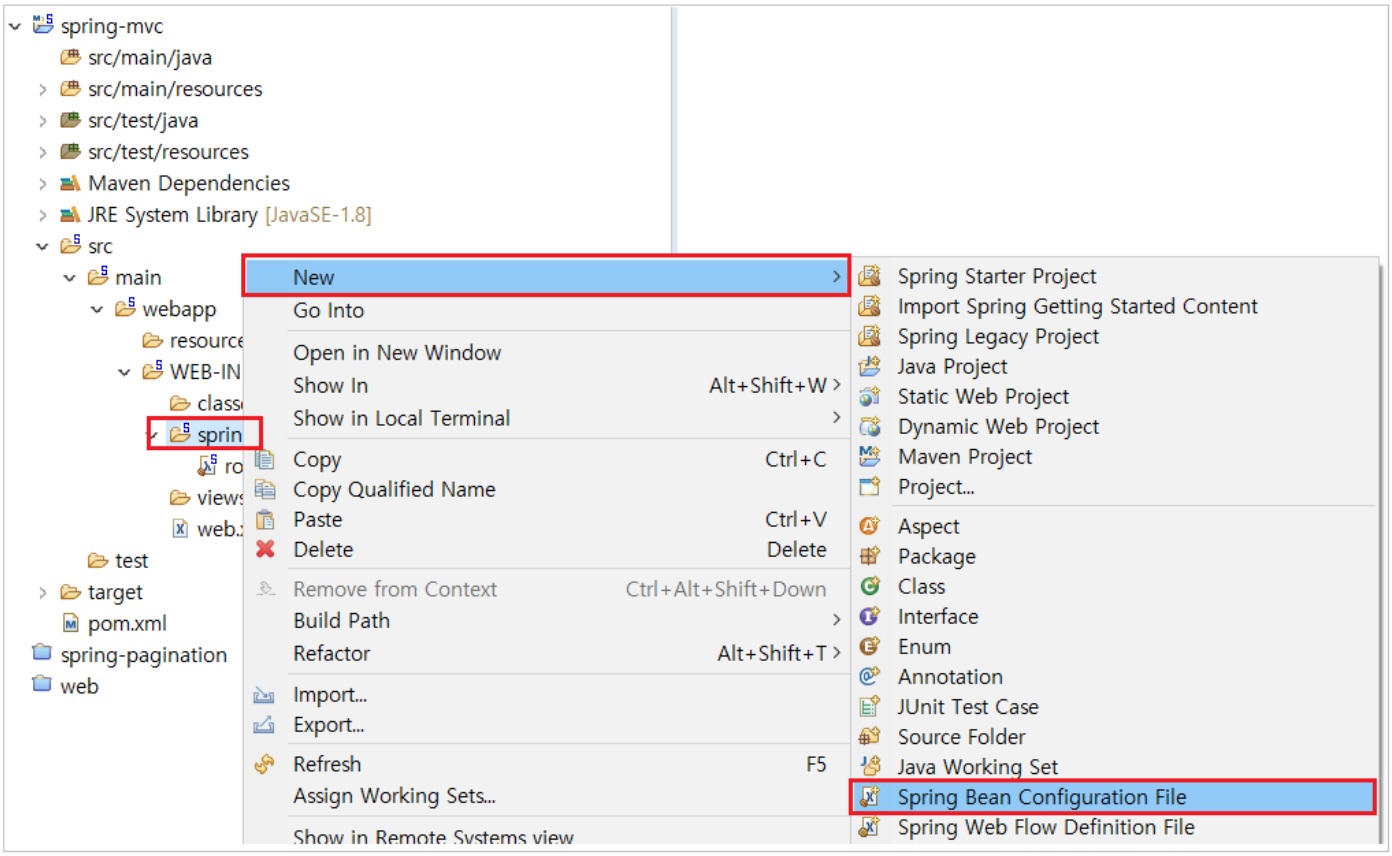
스프링 빈 설정파일(Root 스프링 컨테이너) 네임스페이스 지정하기



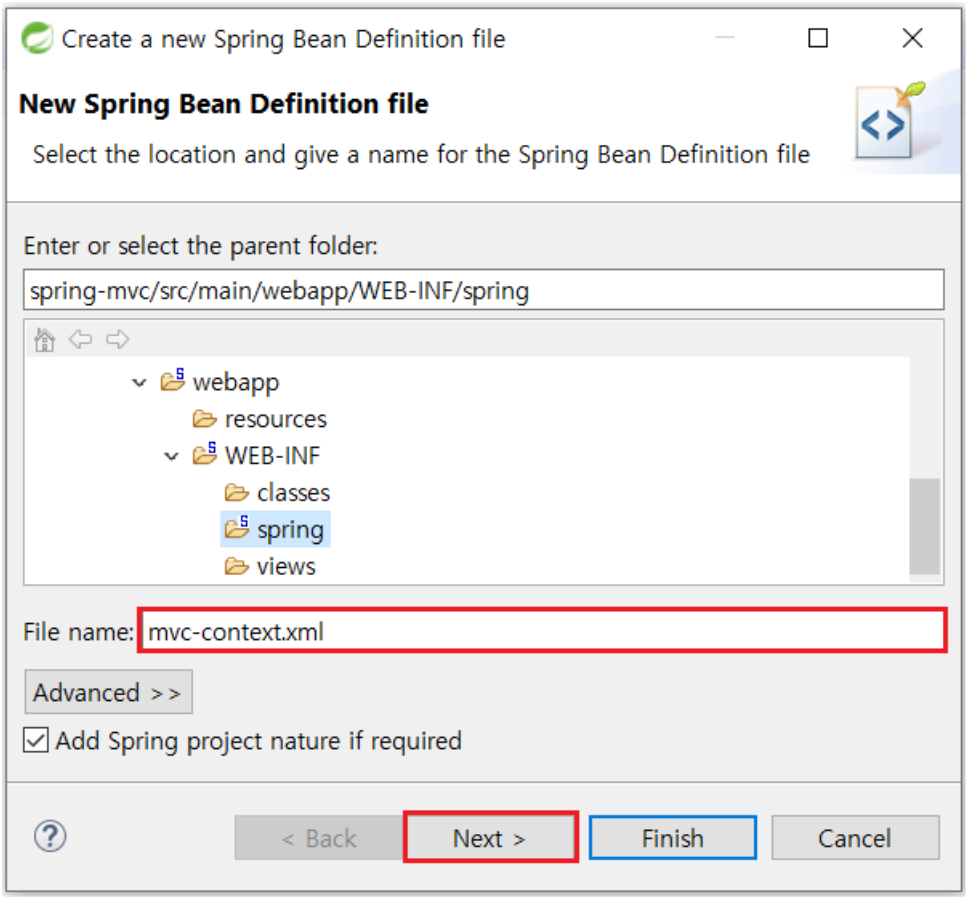
스프링 빈 설정파일(Root 스프링 컨테이너) 생성확인



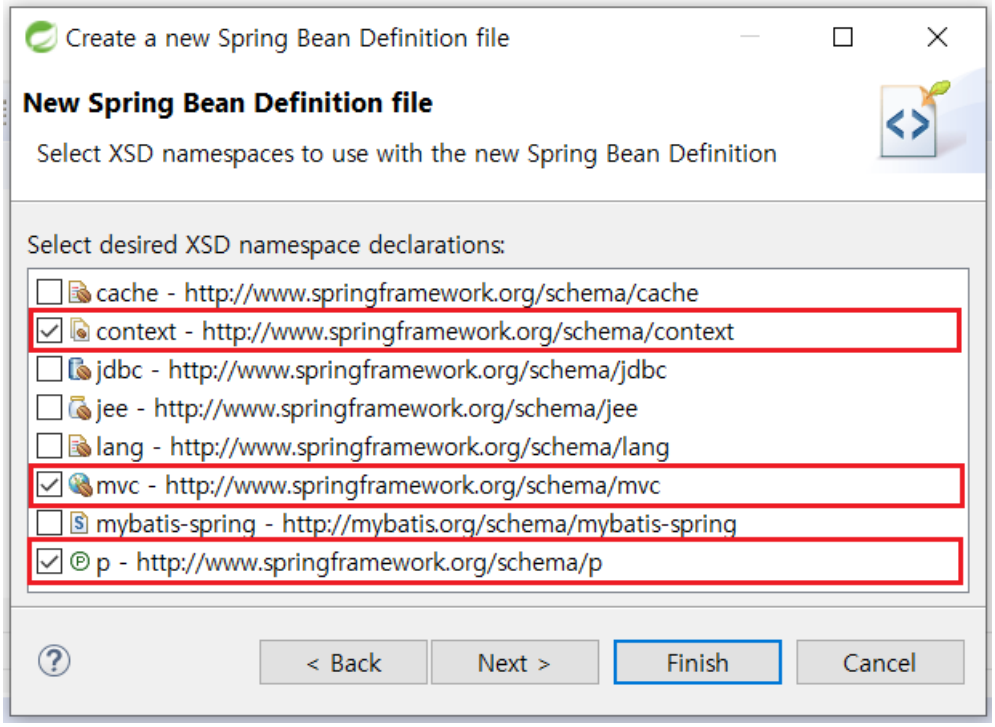
스프링 빈 설정파일(Child 스프링 컨테이너) 생성하기



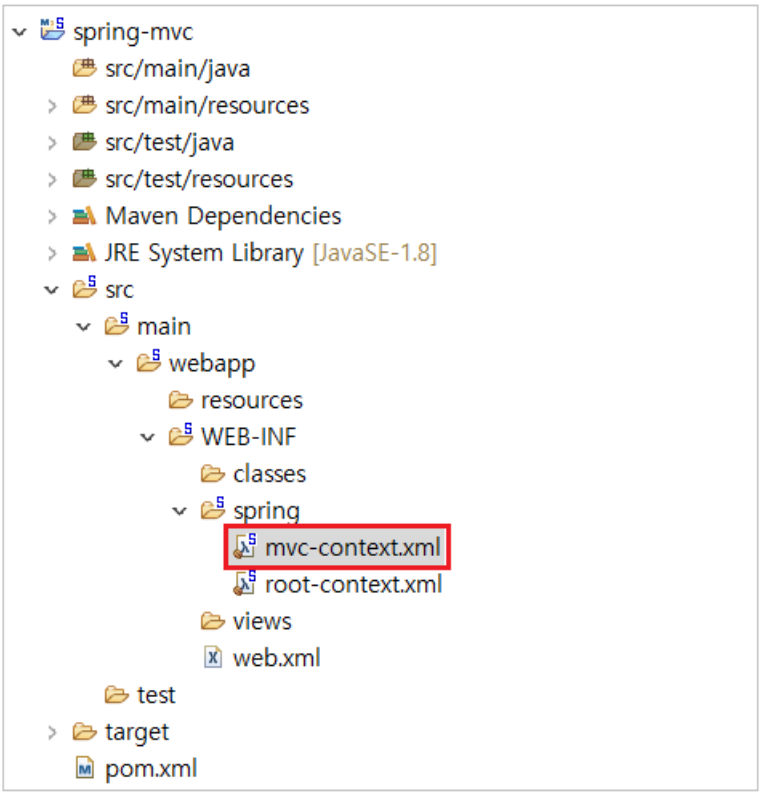
스프링 빈 설정파일(Child 스프링 컨테이너) 이름 정하기



스프링 빈 설정파일(Child 스프링 컨테이너)의 네임스페이스 지정하기



스프링 빈 설정파일(Child 스프링 컨테이너) 생성 확인하기



프로젝트의 web.xml에 스프링 관련 설정정보 추가하기

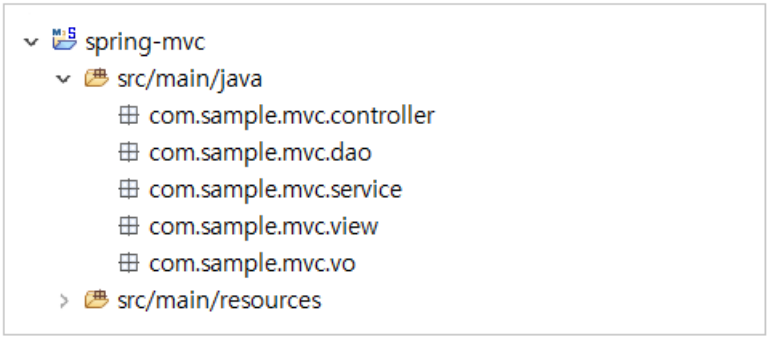
```
<!-- Root Spring Container 설정하기 -->
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/spring/root-context.xml</param-value>
</context-param>
<listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
```

```
<!-- Child Spring Container 설정하기 -->
<servlet>
    <servlet-name>app</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/spring/mvc-context.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>app</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>
```

프로젝트의 web.xml에 필터정보 추가하기

```xml
<!-- 요청 파라미터 한글처리를 위한 필터 등록 -->
<filter>
    <filter-name>characterEncodingFilter</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>characterEncodingFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

패키지 생성하기



HomeController 생성하기

```java
package com.sample.mvc.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class HomeController {

    @GetMapping("/")
    public String home() {

        return "home";
    }
}
```

/webapp/resources에 부트스트랩과 jQuery 라이브러리 추가하기

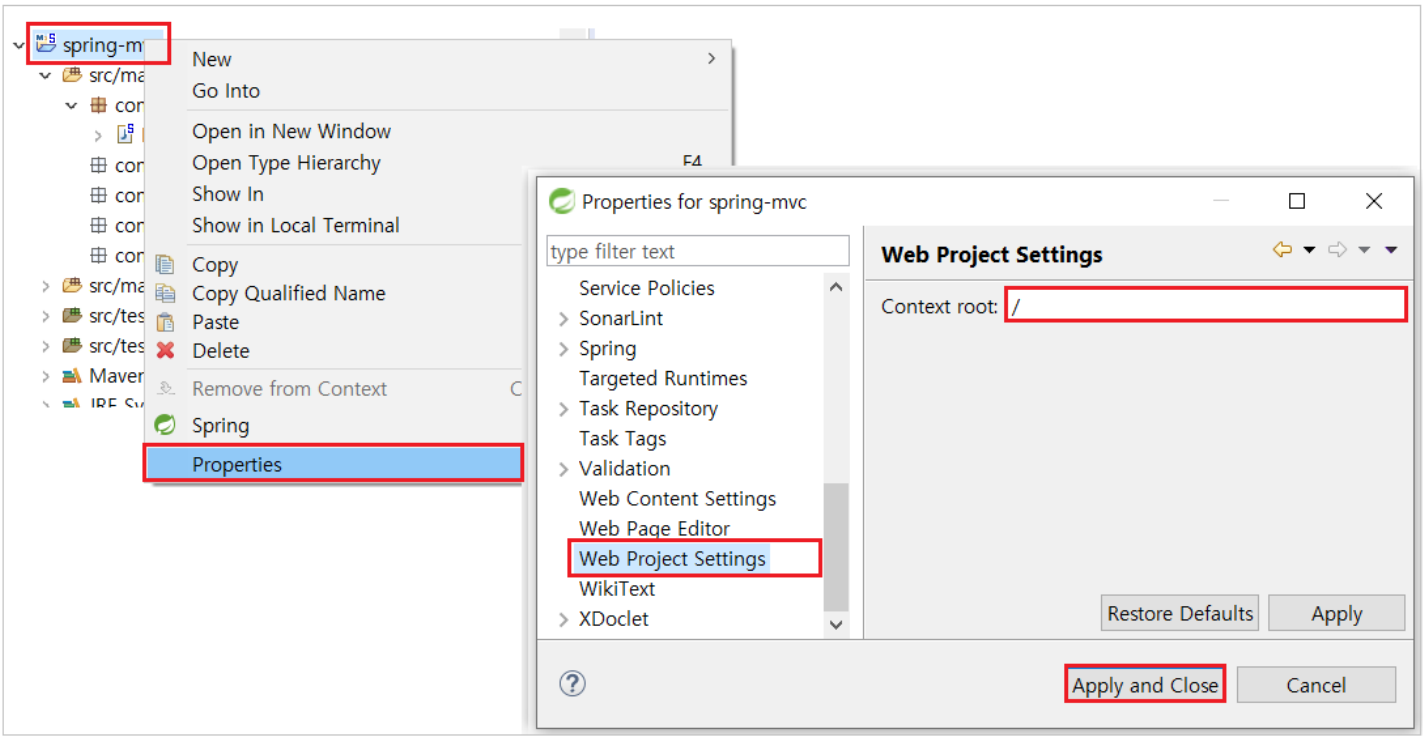
/WEB-INF/views/home.jsp 생성하기

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html lang="ko">
<head>
    <title>Bootstrap Example</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="/resources/bootstrap/css/bootstrap.min.css">
    <script src="/resources/jquery/jquery.min.js"></script>
    <script src="/resources/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="container">
    <div class="page-header">
        <h1>Home Page</h1>
    </div>
</div>

</body>
</html>
```

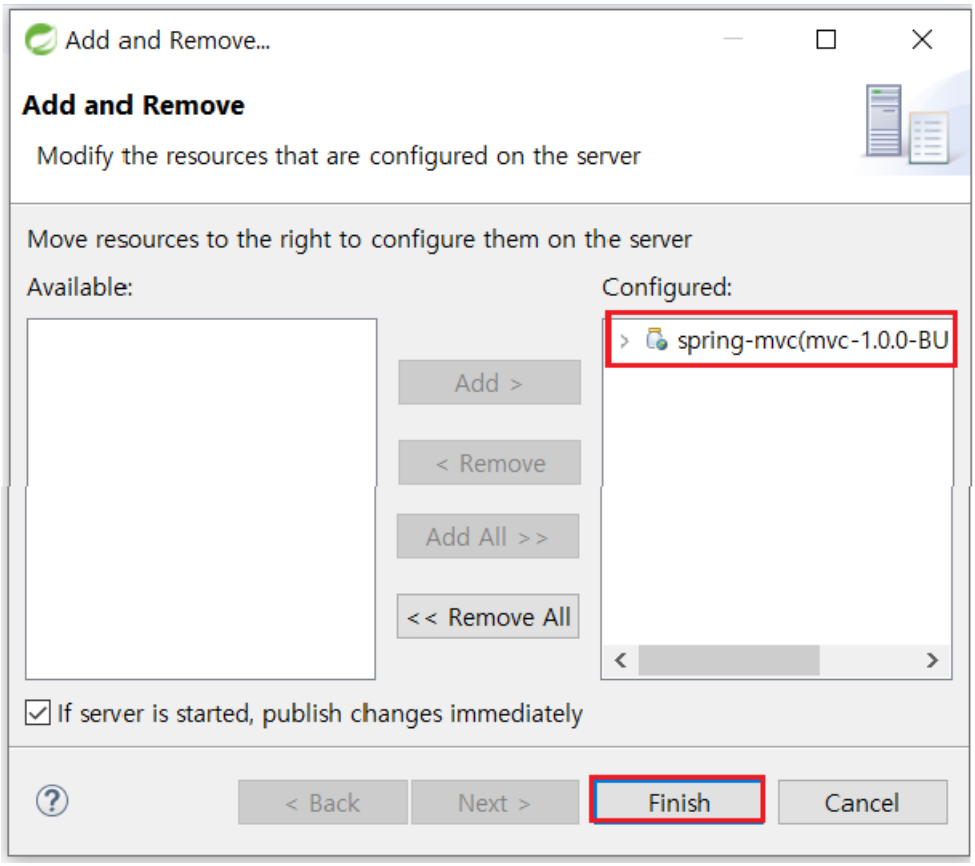
/WEB-INF/spring/mvc-context.xml에 SpringMVC 설정 추가하기

```
<!-- Spring MVC 관련 @어노테이션 설정 정보를 분석하고 적절한 처리를 수행하는 빈을 등록한다-->
<mvc:annotation-driven />
<!-- 정적리소스(css, js, image)는 /resources/**으로 요청하면 /resources/의 리소스를 응답으로 제공한다-->
<mvc:resources mapping="/resources/**" location="/resources/" />

<!-- JSP를 실행해서 동적 컨텐츠(HTML)을 응답으로 제공하는 InternalResourceView를 제공하는 뷰리졸브를 등록한다. -->
<mvc:view-resolvers>
    <!-- 실행할 JSP는 컨트롤러가 반환하는 뷰이름에 /WEB-INF/views/와 .jsp를 붙여서 완성한다. -->
    <mvc:jsp prefix="/WEB-INF/views/" suffix=".jsp"/>
</mvc:view-resolvers>

<!-- com.sample.mvc.controller 패키지에서 클래스를 스캔해서 스프링의 빈으로 등록한다. -->
<context:component-scan base-package="com.sample.mvc.controller" />
```

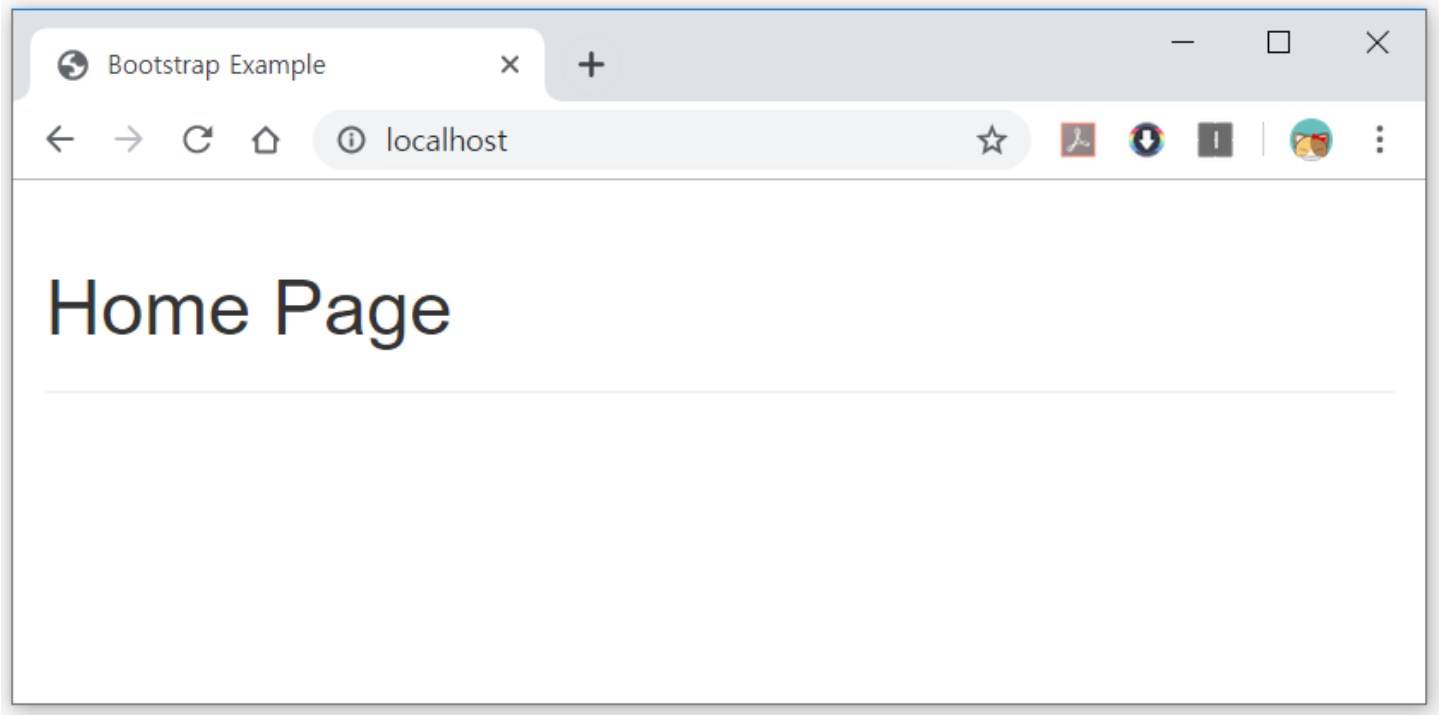
프로젝트의 Context Root를 변경하기



프로젝트를 톰캣에 배포하고 서버를 실행한다.



브라우저에서 시작화면 요청하기



자유게시판 추가하기

자유게시판용 테이블 및 시퀀스 생성하기

```
create table sample_free_board (
    board_no number(7) primary key,
    board_title varchar2(500) not null,
    board_writer varchar2(100) not null,
    board_contents varchar2(2000) not null,
    board_read_count number(6) default 0,
    board_deleted_yn char(1) default 'N',
    board_create_date date default sysdate
);

create sequence free_board_sequence;
```

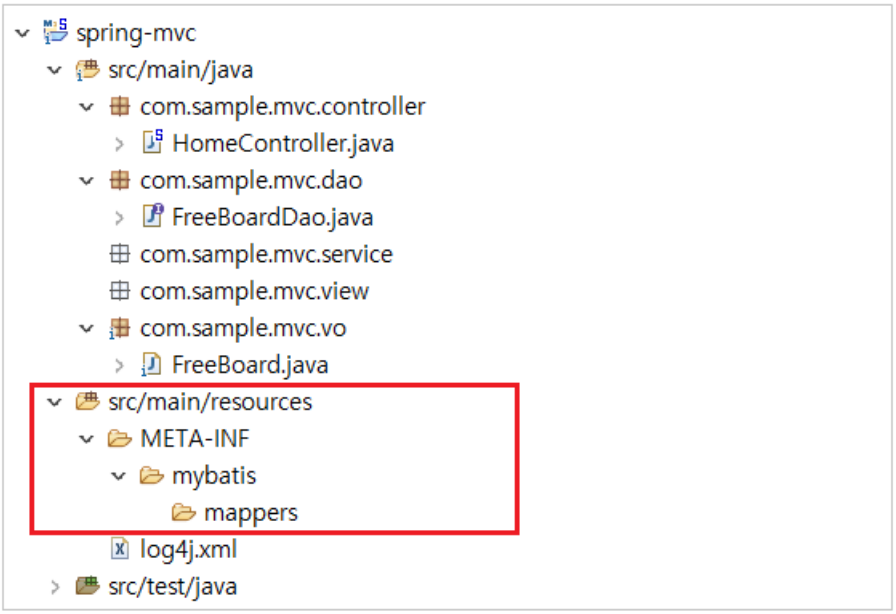
com.sample.mvc.vo 패키지에 Value Object 생성하기

```
package com.sample.mvc.vo;

import java.util.Date;

public class FreeBoard {

    private Integer no;
    private String title;
    private String writer;
    private String contents;
    private Integer readCount;
    private String deletedYn;
    private Date createDate;

    // Getter와 Setter 메소드 추가
}
```

com.sample.mvc.dao 패키지에 매퍼 인터페이스 생성하기

```java
package com.sample.mvc.dao;

import java.util.List;

import com.sample.mvc.vo.FreeBoard;

public interface FreeBoardDao {

    void insertFreeBoard(FreeBoard freeBoard);
    List<FreeBoard> selectAllFreeBoards();
    FreeBoard selectOneFreeBoard(Integer freeBoardNo);
    void updateFreeBoard(FreeBoard freeBoard);
    Integer counts();
}
```

src/main/resources/META-INF 폴더에 마이바티스 환경설정 파일 및 매퍼파일 저장용 폴더 만들기



src/main/resources/META-INF/mybatis 폴더에 마이바티스 환경 설정(mybatis-config.xml) 파일 추가하기

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration  PUBLIC "-//mybatis.org//DTD Configuration 3.0//EN"
    "http://mybatis.org/dtd/mybatis-3-config.dtd">

<configuration>

    <settings>
        <!-- Null값이 허용되는 컬럼에 null값이 입력될 때 NULL값으로 저장되도록 설정한다 -->
        <setting name="jdbcTypeForNull" value="NULL"/>
    </settings>

    <typeAliases>
        <!--
            com.sample.mvc.vo 패키지의 모든 클래스들은 자동으로 타입별칭을 등록하도록 한다
            com.sample.mvc.vo.FreeBoard 인 경우 별칭은 freeBoard로 등록된다.
        -->
        <package name="com.sample.mvc.vo"/>
    </typeAliases>

</configuration>
```

src/main/resources/META-INF/mybatis/mappers 폴더에 매퍼파일(freeboard.xml) 추가하기

```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">

<!-- namespace의 속성값은 FreeBoardDao의 전체경로로 설정한다. -->
<mapper namespace="com.sample.mvc.dao.FreeBoardDao">


</mapper>
```

매퍼파일에 SQL 추가하기

```xml
<!-- 새로운 글 등록하기 -->
<insert id="insertFreeBoard" parameterType="freeBoard">
    insert into sample_free_board
    (board_no, board_title, board_writer, board_contents)
    values
    (free_board_sequence.nextval, #{title}, #{writer}, #{contents})
</insert>
```

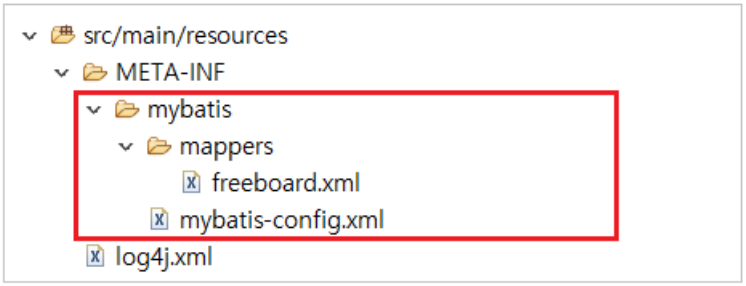
```xml
<!-- 모든 글 조회하기 -->
<select id="selectAllFreeBoards" resultType="freeBoard">
    select
        board_no            as no,
        board_title         as title,
        board_writer        as writer,
        board_contents      as contents,
        board_read_count    as readCount,
        board_deleted_yn    as deletedYn,
        board_create_date   as createDate
    from
        sample_free_board
    where
        board_deleted_yn = 'N'
    order by
        board_no desc
</select>
```

```xml
<!-- 전달받은 글 번호에 해당하는 글 조회하기 -->
<select id="selectOneFreeBoard" parameterType="int" resultType="freeBoard">
    select
        board_no            as no,
        board_title         as title,
        board_writer        as writer,
        board_contents      as contents,
        board_read_count    as readCount,
        board_deleted_yn    as deletedYn,
        board_create_date   as createDate
    from
        sample_free_board
    where
        board_no = #{value}
</select>
```

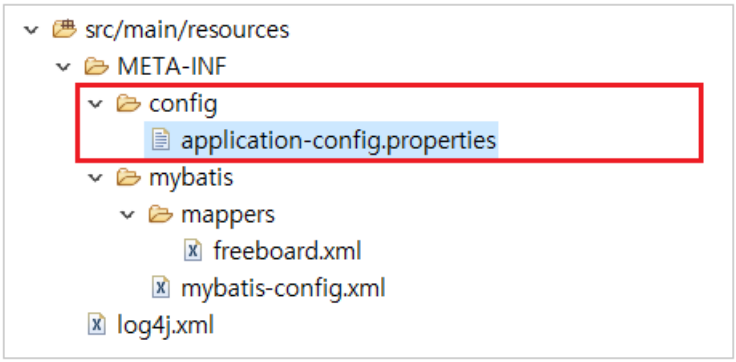
```xml
<!-- 전달받은 글정보로 업데이트 하기 -->
<update id="updateFreeBoard" parameterType="freeBoard">
    update
        sample_free_board
    set
        board_title = #{title},
        board_writer = #{writer},
        board_contents = #{contents},
        board_read_count = #{readCount},
        board_deleted_yn = #{deletedYn}
    where
        board_no = #{no}
</update>
```

```xml
<!-- 모든 글 갯수 조회하기 -->
<select id="counts" resultType="int">
    select
        count(*)
    from
        sample_free_board
    where
        board_deleted_yn = 'N'
</select>
```

마이바티스 설정파일과 매퍼파일 확인하기



src/main/resources/META-INF/config/application-config.properties 파일을 생성하고 DB 연결정보 설정하기




```properties
### Database Connection Information
db.dev.driver=oracle.jdbc.OracleDriver
db.dev.url=jdbc:oracle:thin:@localhost:1521:xe
db.dev.username=hr
db.dev.password=zxcv1234
```

WEB-INF/spring/root-context.xml에 설정정보 추가하기

```xml
<!-- 클래스에 포함된 @어노테이션을 활용해서 객체의 조립을 처리하는 객체가 스프링의 빈으로 등록되게 한다. -->
<context:annotation-config />
<!-- com.sample.mvc.service 패키지에 정의된 클래스를 스캔해서 객체를 생성하고 스프링의 빈으로 등록되게 한다. -->
<context:component-scan base-package="com.sample.mvc.service" />
```

```
<!-- 데이터베이스 연결정보가 설정된 프로퍼티파일을 읽어오는 객체를 스프링의 빈으로 등록되게 한다. -->
<context:property-placeholder location="classpath:/META-INF/config/application-config.properties"/>
<!-- 커넥션풀 객체를 생성해서 스프링의 빈으로 등록되게 한다. -->
<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource"
    p:driverClassName="${db.dev.driver}"
    p:url="${db.dev.url}"
    p:username="${db.dev.username}"
    p:password="${db.dev.password}"/>
<!-- 선언적 트랜잭션 처리를 지원하는 트랜잭션매니저 스프링의 빈으로 등록되게 한다. -->
<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager"
    p:dataSource-ref="dataSource"/>
<!-- 선언적 트랜잭션 처리를 지원받기 위해 @Transactional 어노테이션을 사용할 수 있도록 지원하는 객체를 스프링의 빈으로 등록한다. -->
<tx:annotation-driven transaction-manager="transactionManager"/>
```

```
<!-- 마이바티스의 핵심객체(SqlSessionFactory)를 스프링의 빈으로 등록되게 한다. -->
<bean class="org.mybatis.spring.SqlSessionFactoryBean"
    p:configLocation="classpath:/META-INF/mybatis/mybatis-config.xml"
    p:mapperLocations="classpath:/META-INF/mybatis/mappers/*.xml"/>
<!-- com.sample.mvc.dao 패키지에 정의된 매퍼인터페이스를 스캔해서 매퍼객체를 생성하고 스프링의 빈으로 등록되게 한다. -->
<mybatis-spring:scan base-package="com.sample.mvc.dao"/>
```

자유게시판 관련 서비스 표준 정하기(FreeBoardService 인터페이스 정의하기)

```java
package com.sample.mvc.service;

import java.util.List;

import org.springframework.transaction.annotation.Transactional;

import com.sample.mvc.vo.FreeBoard;

@Transactional
public interface FreeBoardService {

    // 새 글 등록하기
    void addNewFreeBoard(FreeBoard freeBoard);
    // 모든 글 조회하기
    List<FreeBoard> getAllFreeBoards();
    // 지정된 글번호에 해당하는 글 정보 조회하기
    FreeBoard getFreeBoardDetail(Integer freeBoardNo);
    // 지정된 글정보로 변경하기
    void modifyFreeBoard(FreeBoard freeBoard);
    // 글 삭제하기
    void deleteFreeBoard(Integer freeBoardNo);
    // 글 조회수 증가시키기
    void increaseReadCount(Integer freeBoardNo);
}
```

자유게시판 관련 구현서비스 정의하기(FreeBoardServiceImpl 구현 클래스 정의하기)

```java
package com.sample.mvc.service;

import java.util.List;

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

import com.sample.mvc.dao.FreeBoardDao;
import com.sample.mvc.vo.FreeBoard;

@Service
public class FreeBoardServiceImpl implements FreeBoardService {

    @Autowired
    private FreeBoardDao freeBoardDao;

}
```

자유게시판 관련 구현서비스 기능 구현하기(FreeBoardServiceImpl 구현 클래스 구현하기)

```java
// 새 글 등록하기
public void addNewFreeBoard(FreeBoard freeBoard) {
    freeBoardDao.insertFreeBoard(freeBoard);
}

// 모든 글 조회하기
public List<FreeBoard> getAllFreeBoards() {
    return freeBoardDao.selectAllFreeBoards();
}

// 지정된 글번호에 해당하는 글 조회하기
public FreeBoard getFreeBoardDetail(Integer freeBoardNo) {
    return freeBoardDao.selectOneFreeBoard(freeBoardNo);
}
```

```java
// 전달받은 글정보로 변경하기
public void modifyFreeBoard(FreeBoard freeBoard) {
    // 글 번호로 원본글 조회하기
    FreeBoard dest = freeBoardDao.selectOneFreeBoard(freeBoard.getNo());
    // 조회된 원본글에 전달받은 값을 대입하기
    dest.setTitle(freeBoard.getTitle());
    dest.setWriter(freeBoard.getWriter());
    dest.setContents(freeBoard.getContents());
    // 전달받은 글정보로 변경된 글정보를 db에 반영하기
    freeBoardDao.updateFreeBoard(dest);
}
```

```java
// 전달받은 글번호에 해당하는 글 삭제하기
public void deleteFreeBoard(Integer freeBoardNo) {
    // 글 번호로 원본글 조회하기
    FreeBoard dest = freeBoardDao.selectOneFreeBoard(freeBoardNo);
    // 조회된 원본글의 삭제여부를 'Y'로 변경하기
    dest.setDeletedYn("Y");
    // 삭제여부가 'Y'로 변경된 글정보를 db에 반영하기
    freeBoardDao.updateFreeBoard(dest);
}
```

```java
// 전달받은 글번호에 해당하는 글의 조회수 증가시키기
public void increaseReadCount(Integer freeBoardNo) {
    // 글 번호로 원본글 조회하기
    FreeBoard dest = freeBoardDao.selectOneFreeBoard(freeBoardNo);
    // 조회된 원본글의 조회수를 1증가시키기
    dest.setReadCount(dest.getReadCount() + 1);
    // 조회수가 1증가된 글정보를 db에 반영하기
    freeBoardDao.updateFreeBoard(dest);
}
```

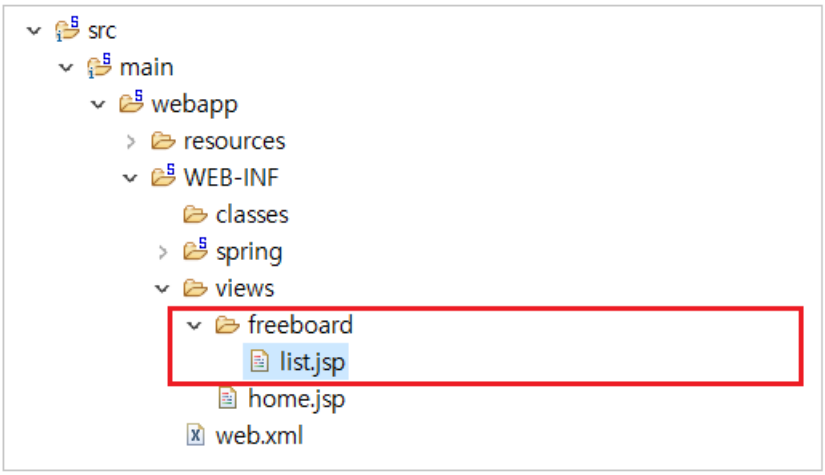
클라이언트의 요청을 처리하는 자유게시판용 컨트롤러 정의하기

```java
package com.sample.mvc.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

import com.sample.mvc.service.FreeBoardService;

@Controller
@RequestMapping("/free")
public class FreeBoardController {

    @Autowired
    private FreeBoardService freeBoardService;

}
```

게시글 조회요청을 처리하는 요청 핸들러 메소드 추가하기

```java
@GetMapping("/list")
public String list(Model model) {
    // 모든 게시글을 조회한다.
    List<FreeBoard> freeBoards = freeBoardService.getAllFreeBoards();
    // 조회된 게시글을 JSP에 전달하기 위해서 Model에 저장한다.
    model.addAttribute("freeBoards", freeBoards);

    // 응답을 제공할 jsp페이지는 /WEB-INF/views/freeboard/list.jsp다.
    return "freeboard/list";
}
```

게시글 목록을 제공하는 jsp페이지 정의하기(/WEB-INF/views/freeboard/list.jsp 정의하기)

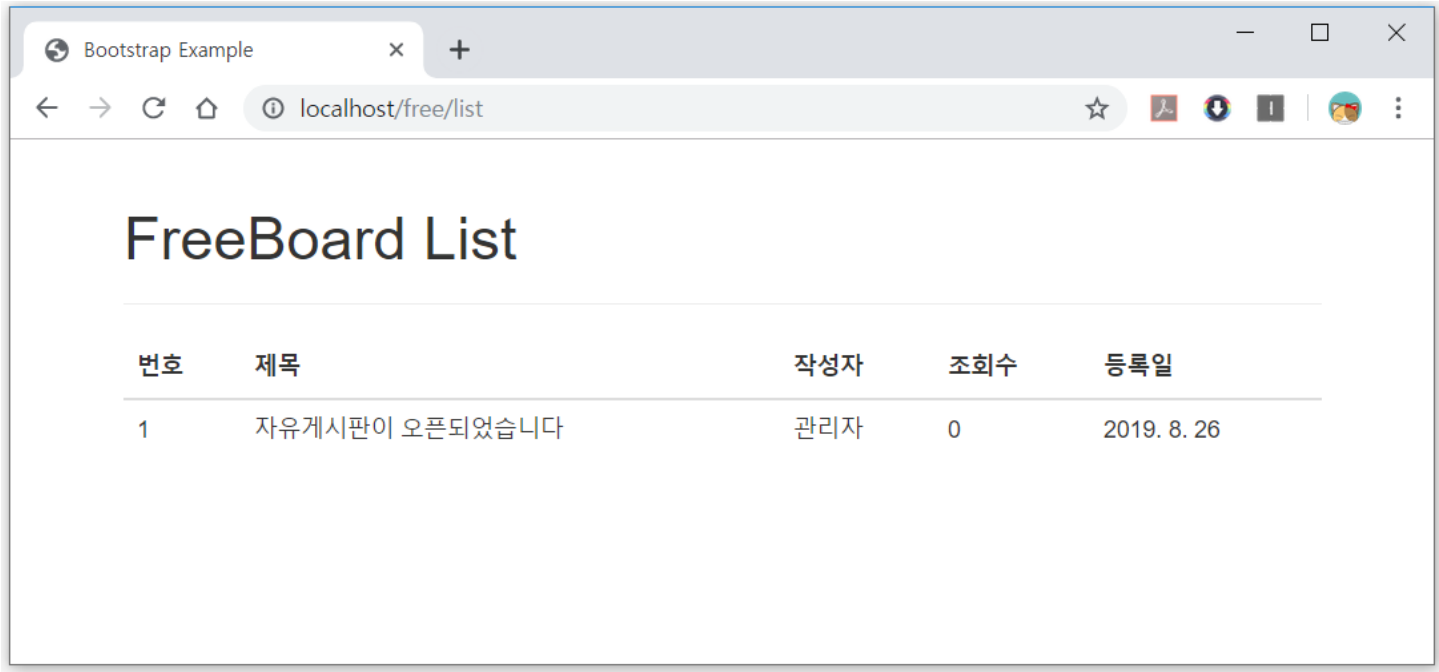


list.jsp 페이지 구현하기

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!-- 태그 라이브러리 추가하기 -->
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<!DOCTYPE html>
<html lang="ko">
<head>
    <title>Bootstrap Example</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="/resources/bootstrap/css/bootstrap.min.css">
    <script src="/resources/jquery/jquery.min.js"></script>
    <script src="/resources/bootstrap/js/bootstrap.min.js"></script>
</head>
<body>

<div class="container">
    <div class="page-header">
        <h1>FreeBoard List</h1>
    </div>
```

```
    <div class="row">
        <div class="col-xs-12">
            <table class="table">
                <thead>
                    <tr>
                        <th>번호</th>
                        <th>제목</th>
                        <th>작성자</th>
                        <th>조회수</th>
                        <th>등록일</th>
                    </tr>
                </thead>
                <tbody>
                <!-- Model에 저장해둔 게시글 표시하기 -->
                <c:forEach var="freeBoard" items="${freeBoards }">
                    <tr>
                        <td>${freeBoard.no }</td>
                        <td>${freeBoard.title }</td>
                        <td>${freeBoard.writer }</td>
                        <td>${freeBoard.readCount }</td>
                        <td><fmt:formatDate value="${freeBoard.createDate }"/></td>
                    </tr>
                </c:forEach>
                </tbody>
            </table>
        </div>
    </div>
</div>

</body>
</html>
```

브라우저에서 자유게시판 목록을 요청한다.



(샘플 게시글을 오라클에 미리 등록한 다음 조회한 결과입니다.)