

The logo for the Spring Framework, featuring the text "Spring Framework" in a bold, dark blue font, centered within a white rectangular box. The box is framed by a thin, light blue border. The word "Spring" is on the top line and "Framework" is on the bottom line.

Spring Framework

▶ Spring Framework

자바 플랫폼을 위한 오픈 소스 애플리케이션 프레임워크로
간단하게 스프링(Spring)이라고도 부름
동적인 웹 사이트를 개발하기 위한 여러 가지 서비스를 제공하고 있으며
대한민국 공공기간의 웹 서비스 개발 시 사용을 권장하고 있는
전자정부 프레임워크의 기반 기술로 쓰임

* Spring 공식 사이트 : <https://spring.io/>

▶ Spring Framework

✓ 특징

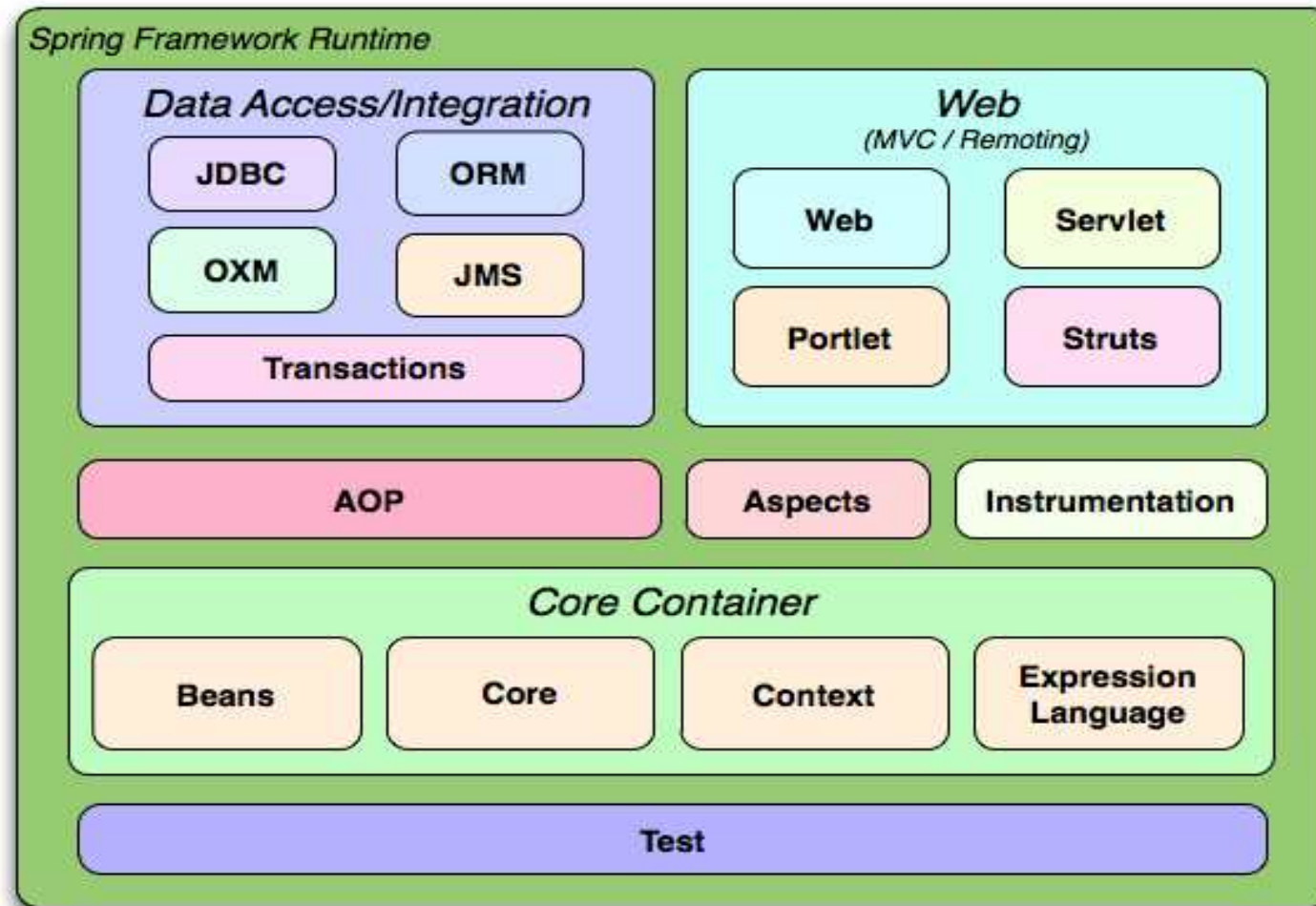
DI (Dependency Injection) 의존성 주입	설정 파일이나 어노테이션을 통해 객체 간의 의존 관계를 설정하여 개발자가 직접 의존하는 객체를 생성할 필요 없음
Spring AOP (Aspect Oriented Programming) 관점 지향 프로그래밍	트랜잭션, 로깅, 보안 등 여러 모듈, 여러 계층에서 공통으로 필요로 하는 기능의 경우 해당 기능들을 분리하여 관리
POJO (Plain Old Java Object)	일반적인 J2EE 프레임워크에 비해 특정 라이브러리를 사용할 필요가 없어 개발이 쉬우며 기존 라이브러리의 지원 용이
IoC (Inversion of Control) 제어 반전	컨트롤의 제어권이 개발자가 아니라 프레임워크에 있다는 뜻으로 객체의 생성부터 모든 생명주기의 관리까지 프레임워크가 주도 객체를 생성하고 직접 호출하는 프로그램이 아니라 만들어둔 자원을 호출하여 사용

▶ Spring Framework

✓ 특징

Spring JDBC	MyBatis나 Hibernate 등의 데이터베이스를 처리하는 영속성 프레임워크와 연결할 수 있는 인터페이스 제공
Spring MVC	MVC 디자인 패턴을 통해 웹 애플리케이션의 Model, View, Controller 사이의 의존 관계를 DI 컨테이너에서 개발자가 아닌 서버가 객체들을 관리하는 웹 애플리케이션 구축
PSA (Portable Service Abstraction)	스프링을 다른 여러 모듈을 사용함에 있어 별도의 추상화 레이어 제공 예를 들어 JPA를 사용할 때 Spring JPA를 사용하여 추상화하므로 실제 구현에 있어 Hibernate를 사용하든 EclipseLink를 사용하든 개발자는 모듈의 의존 없이 프로그램에 집중할 수 있음

▶ Spring의 구성 모듈



▶ Spring의 구성 모듈

✓ Data 접근 계층

JDBC나 데이터베이스에 연결하는 모듈로 Data 트랜잭션에 해당하는 기능을 담당하여 영속성 프레임워크의 연결 담당

✓ MVC 계층(MVC / Remoting)

Spring Framework에서 Servlet, Struts 등 웹 구현 기술과의 연결점을 Spring MVC 구성으로 지원하기 위해 제공되는 모듈 계층
또한 스프링의 리모팅 기술로 RMI, Hessian, Burlap, JAX-WS, Http 호출자 그리고 REST API 모듈 제공

▶ Spring의 구성 모듈

✓ AOP 계층

Spring에서 각 흐름 간 공통된 코드를 한 쪽으로 빼내어 필요한 시점에 해당 코드를 첨부하게 하기 위해 지원하는 계층
별도의 proxy를 두어 동작, 이를 통해 객체 간의 결합도를 낮출 수 있음

✓ Core Container

Spring의 핵심 부분이라고 할 수 있으며 모든 스프링 관련 모듈은 이 Core Container 기반으로 구축
Spring의 근간이 되는 IoC(또는 DI) 기능을 지원하는 영역 담당
BeanFactory를 기반으로 Bean클래스들을 제어할 수 있는 기능 지원

▶ Spring의 구성 모듈

✓ 모듈 정리

모듈 명	내용
spring-beans	스프링 컨테이너를 이용해서 객체를 생성하는 기본 기능 제공
spring-context	객체 생성, 라이프 사이클 처리, 스키마 확장 등의 기능 제공
spring-aop	AOP 기능 제공
spring-web	REST클라이언트 데이터 변환 처리, 서블릿 필터, 파일 업로드 지원 등 웹 개발에 필요한 기반 기능 제공
spring-webmvc	스프링 기반의 MVC 프레임워크, 웹 애플리케이션을 개발하는데 필요한 컨트롤러, 뷰 구현 제공
spring-websocket	스프링 MVC에서 웹 소켓 연동을 처리할 수 있도록 제공

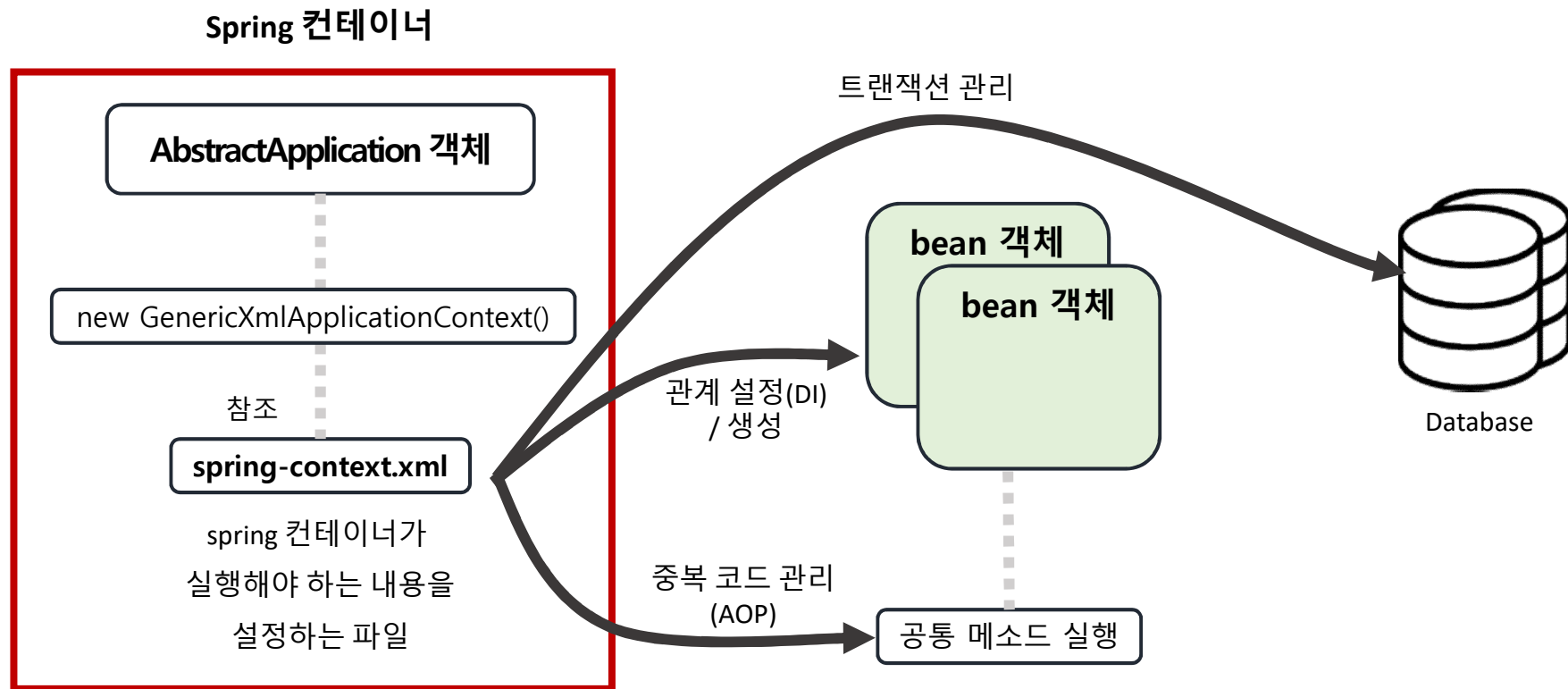
▶ Spring의 구성 모듈

✓ 모듈 정리

모듈 명	내용
spring-oxm	XML과 자바 객체 간의 매핑을 처리하기 위한 API 제공
spring-tx	트랜잭션 처리를 위한 추상 레이어 제공
spring-jdbc	JDBC프로그래밍을 보다 쉽게 할 수 있는 템플릿 제공
spring-orm	Hibernate, JPA, MyBatis 등과의 연동 지원
spring-jms	JMS서버와 메시지를 쉽게 주고 받을 수 있도록 하기 위한 템플릿
spring-context-support	스케줄링, 메일발송, 캐시연동, 벨로시티 등 부가기능제공

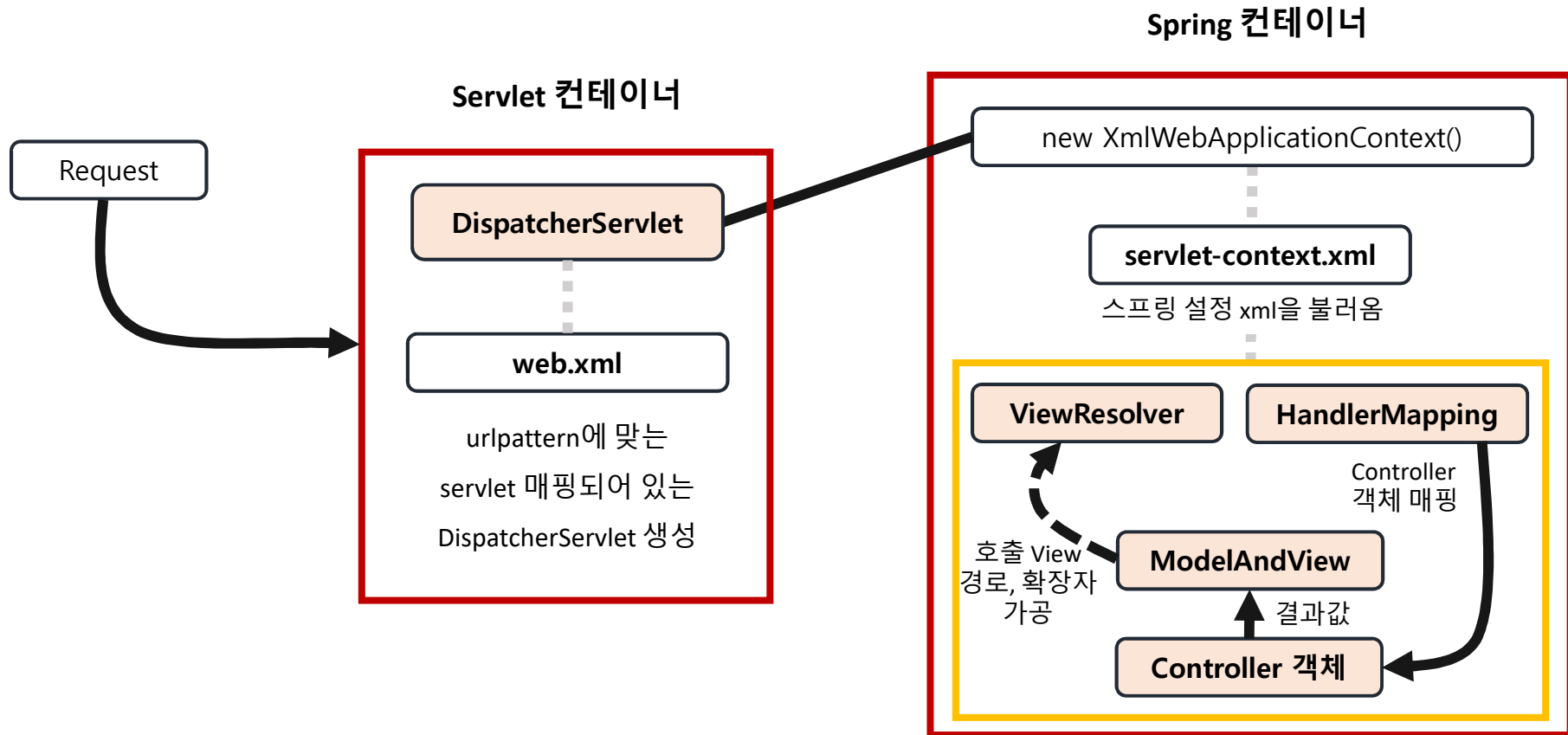
▶ Spring의 동작 구조

✓ Spring 애플리케이션



▶ Spring의 동작 구조

✓ Spring 웹



▶ Spring의 동작 구조

✓ XML파일

Spring컨테이너 구동 시 한 개의 spring환경 설정된 xml파일을 불러오는데 이 파일에 bean, aop, transaction 등 여러 사항을 다 작성하여 구동

```
<?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:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="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-4.2.xsd">

    <!-- controllerMapping -->

    <bean class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
        <property name="mappings">
            <props>
                <prop key="/Login.do">login</prop>
                <prop key="/board.do">board</prop>
            </props>
        </property>
    </bean>
    <bean id="login" class="com.kh.mvc2.user.controller.LoginController"></bean>
    <bean id="board" class="com.kh.mvc2.board.controller.BoardController"></bean>

    <!-- viewResolver -->

    <bean id="viewResolver"
          class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/board/"></property>
        <property name="suffix" value=".jsp"></property>
    </bean>
```

▶ Spring의 동작 구조

✓ @Annotation

xml파일에는 구동 시킬 필수 요소만 작성하고 소스 코드에 Annotation으로 표시하여 구동

```
<?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:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="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-4.2.xsd">
    <!-- 어노테이션 적용 -->
    <context:component-scan base-package="com.kh.mvc2"></context:component-scan>
</beans>
```

xml 파일

```
package com.kh.mvc2.board.controller;

import java.util.List;

@Controller
public class BoardController {
    @RequestMapping(value="/board.do", method=RequestMethod.GET)
    public ModelAndView getListGet(HttpServletRequest request, HttpServletResponse response) {
        System.out.println("get");
        BoardVo board=new BoardVo();
        List<BoardVo> list=new BoardDAO().getList();

        ModelAndView mv=new ModelAndView();
        mv.addObject("boards",list);
        mv.setViewName("/WEB-INF/board/boardlist.jsp");

        return mv;
    }
    @RequestMapping(value="/board.do", method=RequestMethod.POST)
    public ModelAndView getListPost(HttpServletRequest request, HttpServletResponse response) {
        System.out.println("Post");
        BoardVo board=new BoardVo();
        List<BoardVo> list=new BoardDAO().getList();

        ModelAndView mv=new ModelAndView();
        mv.addObject("boards",list);
        mv.setViewName("/WEB-INF/board/boardlist.jsp");

        return mv;
    }
}
```

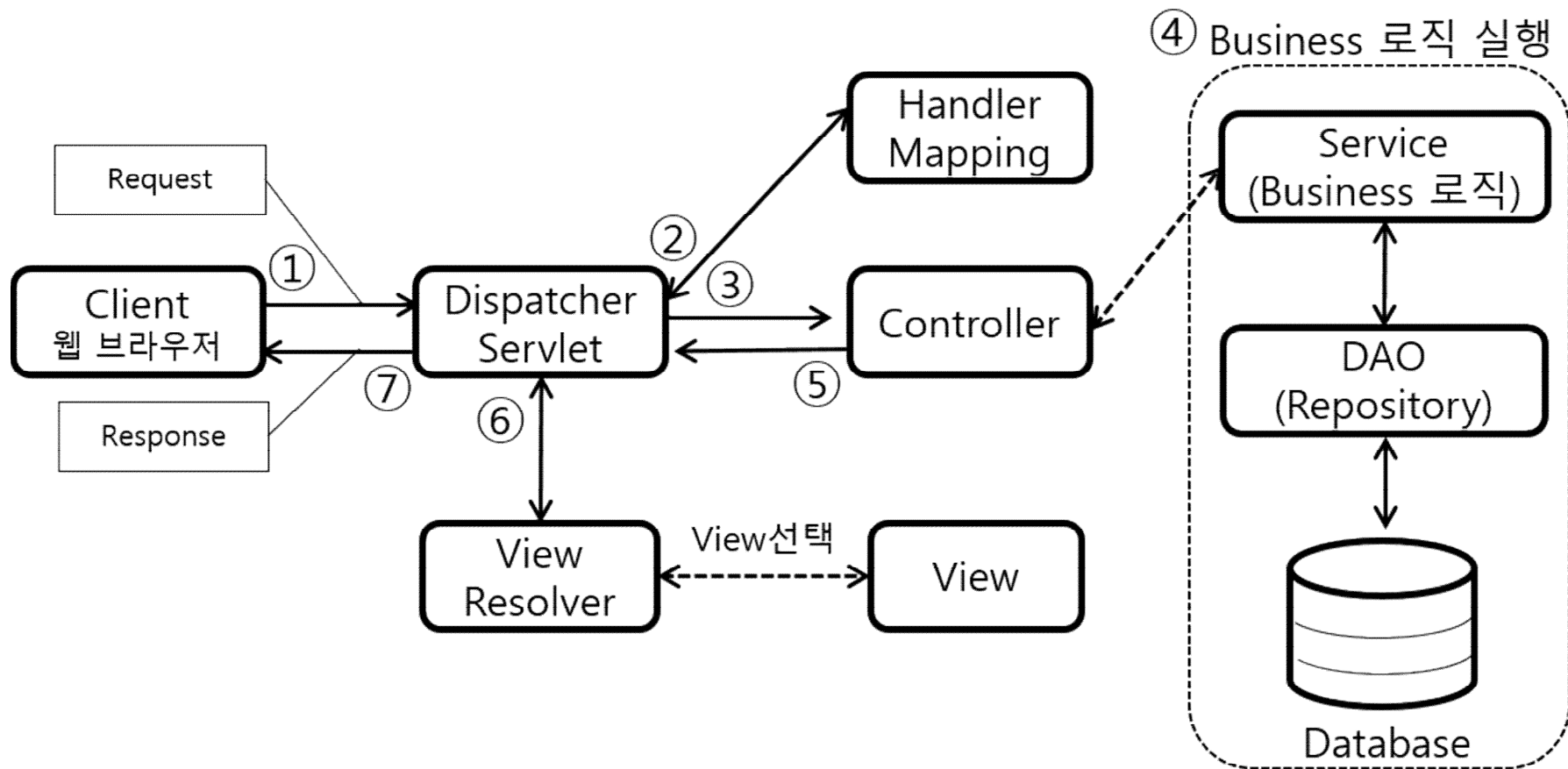
소스 코드

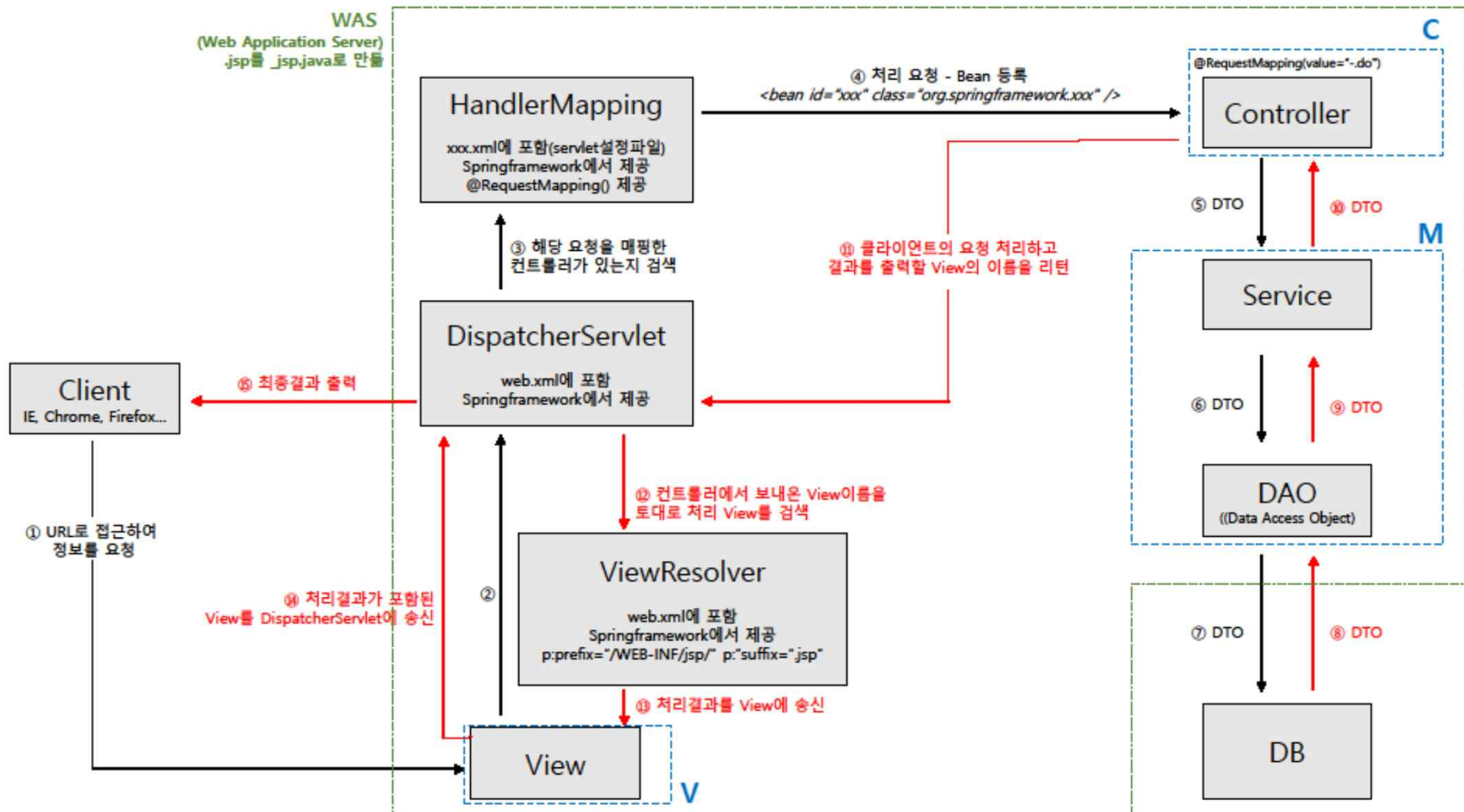
▶ Spring MVC

Spring Framework에서는 클라이언트의 화면을 표현하기 위한 View와 서비스를 수행하기 위한 개발 로직 부분을 나누는 MVC2 패턴 지원
또한 Model, View, Controller 사이의 의존 관계를 DI 컨테이너에서 관리하여 유연한 웹 애플리케이션을 쉽게 구현 및 개발 가능

▶ Spring MVC

✓ Spring MVC 요청 처리 과정





-
- 
1. 웹 애플리케이션이 실행되면 Tomcat(WAS)에 의해 web.xml이 loading된다.
 2. web.xml에 등록되어 있는 ContextLoaderListener(Java Class)가 생성된다. ContextLoaderListener 클래스는 ServletContextListener 인터페이스를 구현하고 있으며, ApplicationContext를 생성하는 역할을 수행한다.
 3. 생성된 ContextLoaderListener는 root-context.xml을 loading한다.
 4. root-context.xml에 등록되어 있는 Spring Container가 구동된다. 이 때 개발자가 작성한 비즈니스 로직에 대한 부분과 DAO, VO 객체들이 생성된다.
 5. 클라이언트로부터 웹 애플리케이션이 요청이 온다.
 6. DispatcherServlet(Servlet)이 생성된다. DispatcherServlet은 FrontController의 역할을 수행한다. 클라이언트로부터 요청 온 메시지를 분석하여 알맞은 PageController에게 전달하고 응답을 받아 요청에 따른 응답을 어떻게 할 지 결정만한다. 실질적인 작업은 PageController에서 이루어지기 때문이다. 이러한 클래스들을 HandlerMapping, ViewResolver 클래스라고 한다.
 7. DispatcherServlet은 servlet-context.xml을 loading 한다.
 8. 두번째 Spring Container가 구동되며 응답에 맞는 PageController 들이 동작한다. 이 때 첫번째 Spring Container 가 구동되면서 생성된 DAO, VO, ServiceImpl 클래스들과 협업하여 알맞은 작업을 처리하게 된다.
-

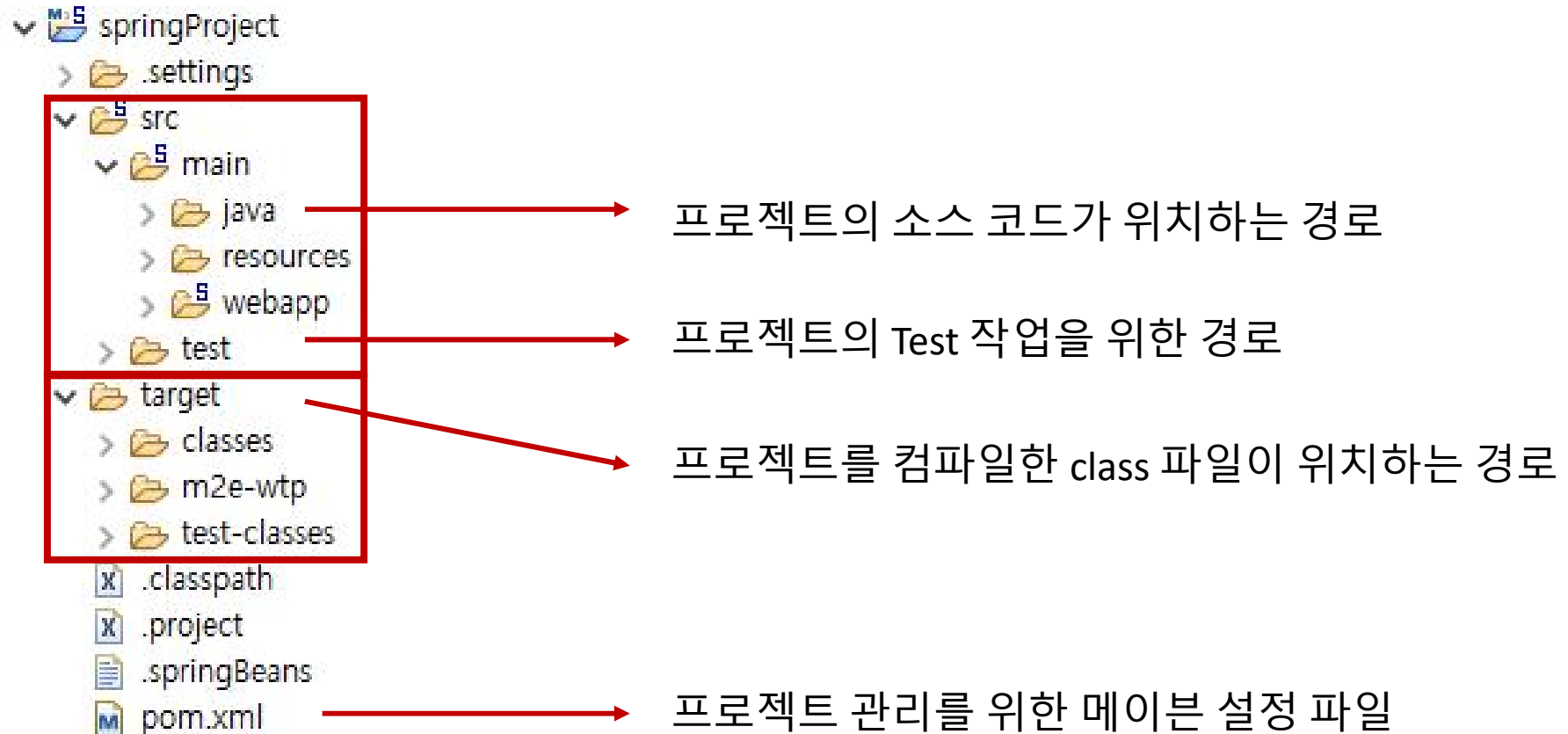
▶ Spring MVC

✓ Spring MVC 구성 요소

구성 요소	설명
DispatcherServlet	클라이언트의 요청(Request)을 전달 받고 요청에 맞는 컨트롤러가 리턴한 결과 값을 View에 전달하여 알맞은 응답(Response) 생성
HandlerMapping	클라이언트의 요청 URL을 어떤 컨트롤러가 처리할 지 결정
Controller	클라이언트의 요청을 처리한 뒤 결과를 DispatcherServlet에게 리턴
ModelAndView	컨트롤러가 처리한 결과 정보 및 뷰 선택에 필요한 정보를 담음
ViewResolver	컨트롤러의 처리 결과를 생성할 View 결정
View	컨트롤러의 처리 결과 화면 생성, JSP나 Velocity 템플릿 파일 등을 View로 사용

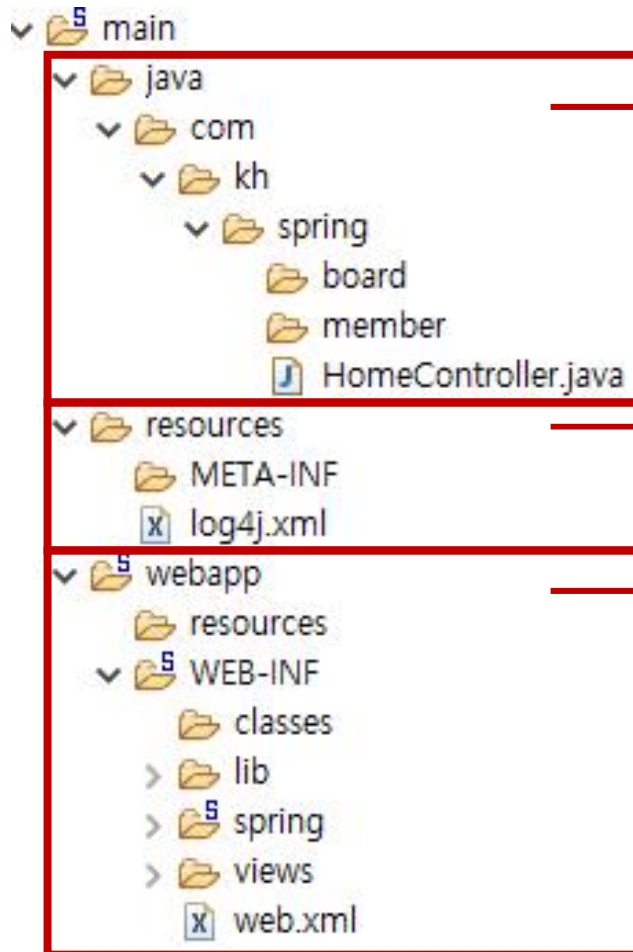
▶ Spring 프로젝트 구조

✓ Spring 프로젝트 폴더 구조



▶ Spring 프로젝트 구조

✓ main 폴더



java

우리가 작성하는 .java 파일의 위치

resources

프로젝트 설정에 필요한 xml 등의 설정파일들

webapp

사용자 화면에 표시할 view 관련 파일들과
웹 컨테이너 설정에 필요한 xml 파일들

▶ Spring 프로젝트 구조

✓ webapp 폴더

