5장. 스프링 MVC 구조

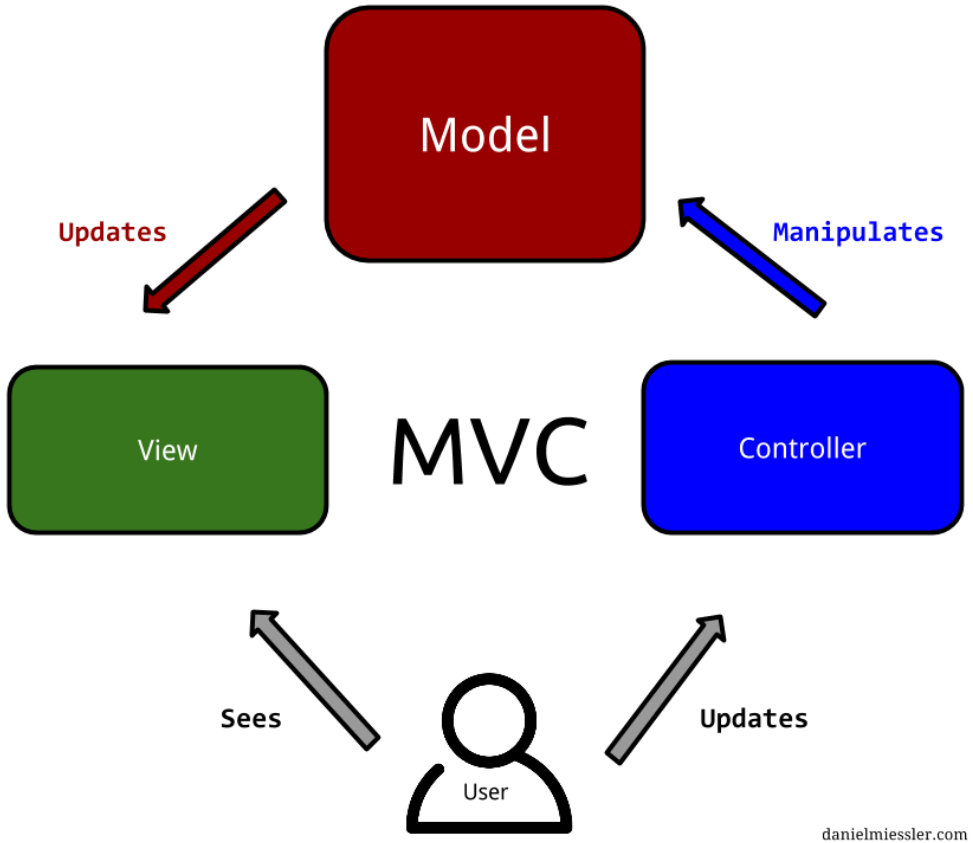
동의과학대학교
컴퓨터정보과
김진숙

학습목표

- MVC구조의 이해
- 스프링 MVC의 다양한 예제의 학습

MVC(Model-View-Controller)

- 대부분의 서블릿 기반 프레임워크들이 사용하는 방식
- 데이터와 처리, 화면을 분리하는 방식
- 웹에서는 Model 2 방식으로 표현



스프링과 스프링 MVC

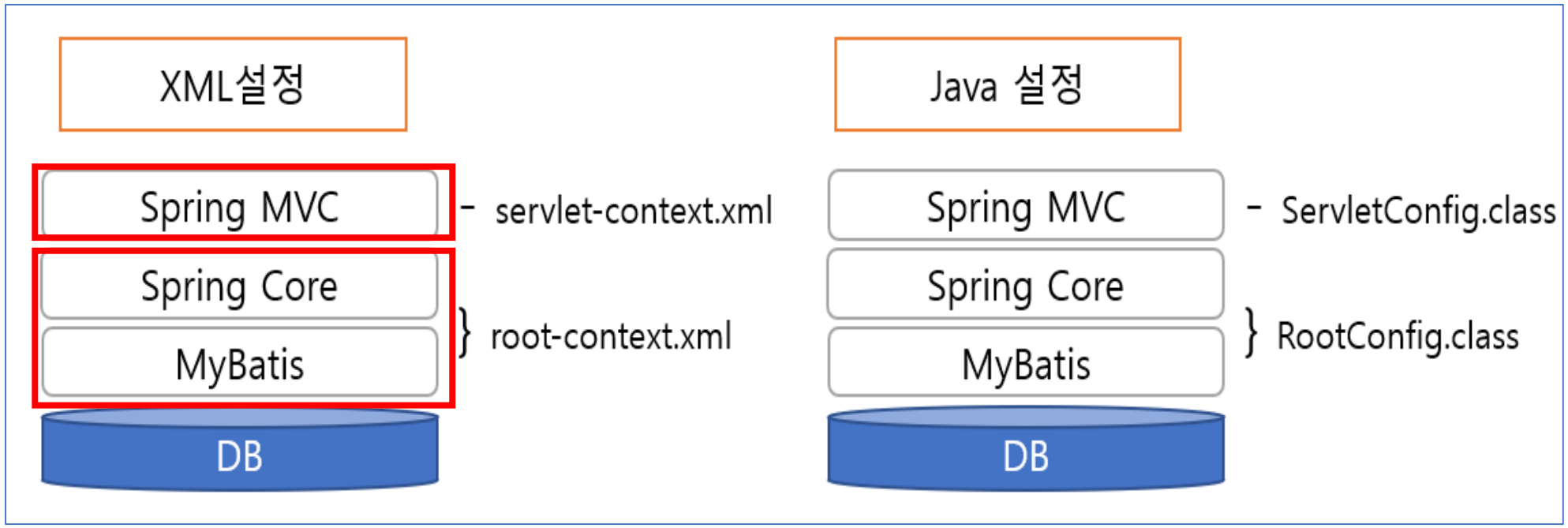
- 스프링 프레임워크 Core + 여러 Sub 프로젝트들
- https://spring.io/projects

- 별도로 결합해서 사용하기 때문에 설정 역시 별도로 처리 가능

서블릿(모델2)과 스프링 MVC 컨트롤러

서블릿	스프링 컨트롤러
• 클래스에서 URL 분기	• 클래스 또는 **메서드** 단위에서 URL 분기 가능
• GET/POST 메서드 선택과 오버라이드	• @GetMapping/ @PostMapping
• 고정된 파라미터와 리턴 타입 (HttpServletRequest, HttpServletResponse)	• 파라미터 자동 수집 • 상황에 따른 반환 타입 조정
• 수동으로 forward	• 자동으로 forward
• 의존성 주입 불가	• 스프링을 통해 객체 주입 가능
• JSON 처리 불편	• @RestController 통한 자동 처리
	• 어노테이션을 통한 많은 기능 지원
	• 다양한 뷰 처리 기능

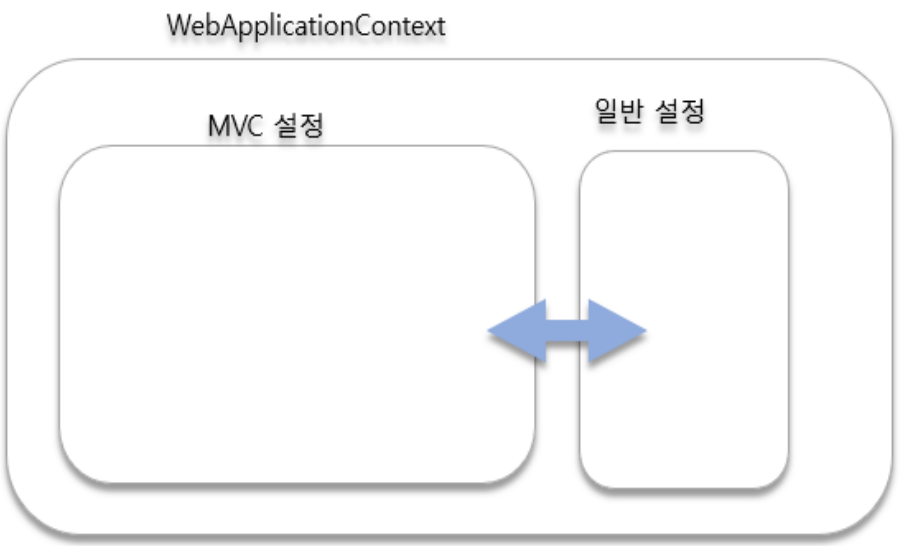
일반적인 스프링 + 스프링 MVC

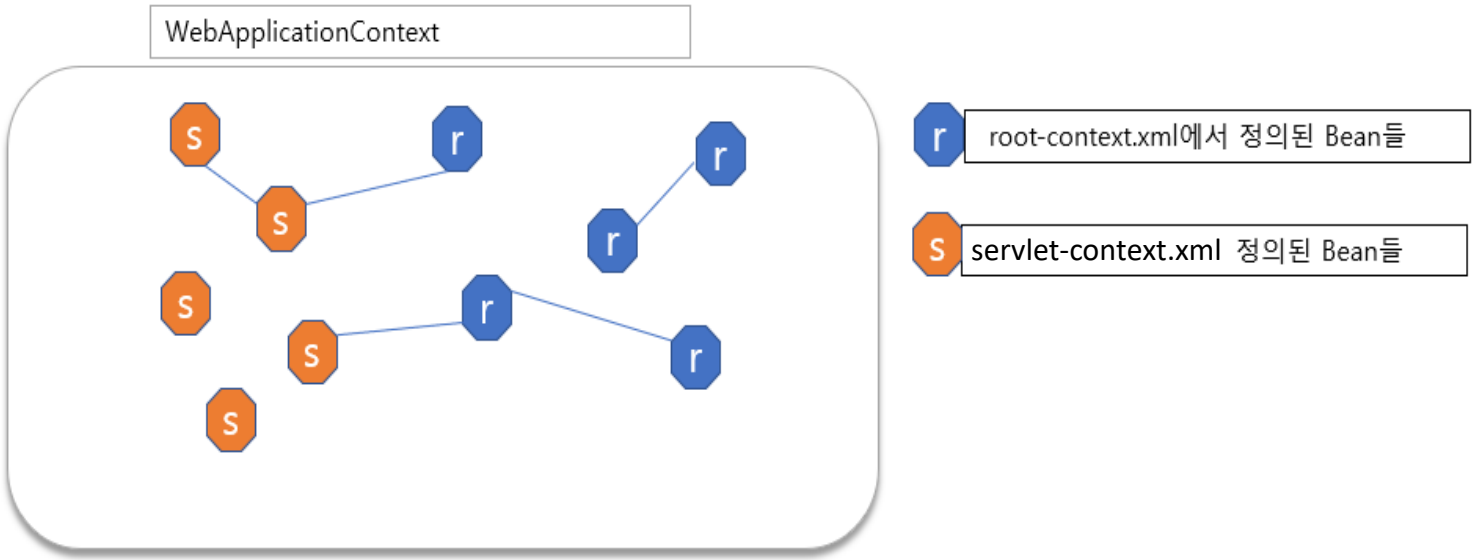


* XML이나 Java설정 이용시에 설정 분리

웹 프로젝트의 구조

- 스프링을 실행하는 존재
  - ApplicationContext => WebApplicationContext
  - **root-context.xml** : 일반 Java(POJO) 영역
  - **servlet-context.xml** : 웹 관련 영역
  - 두 영역은 다음과 같이 연동되는 방식으로 동작하기 때문에 설정을 분리해도 통합해서 사용가능



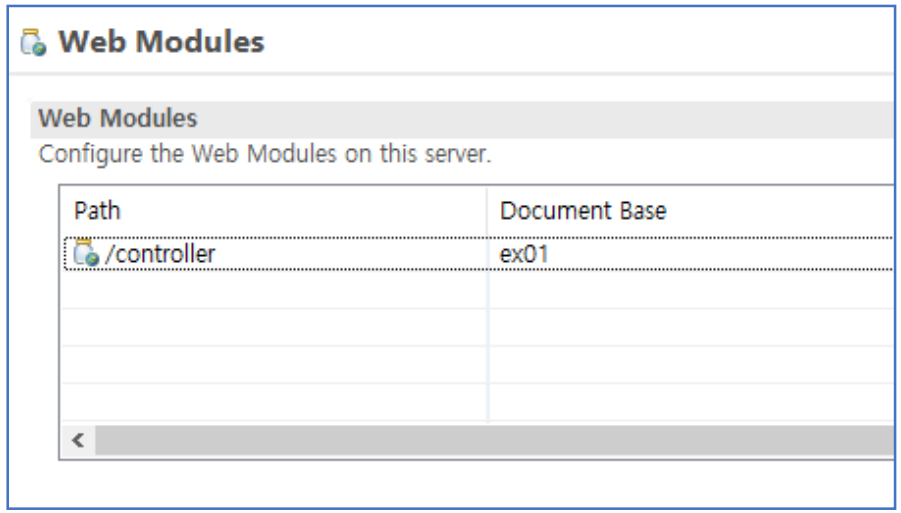


[실습] - ex01

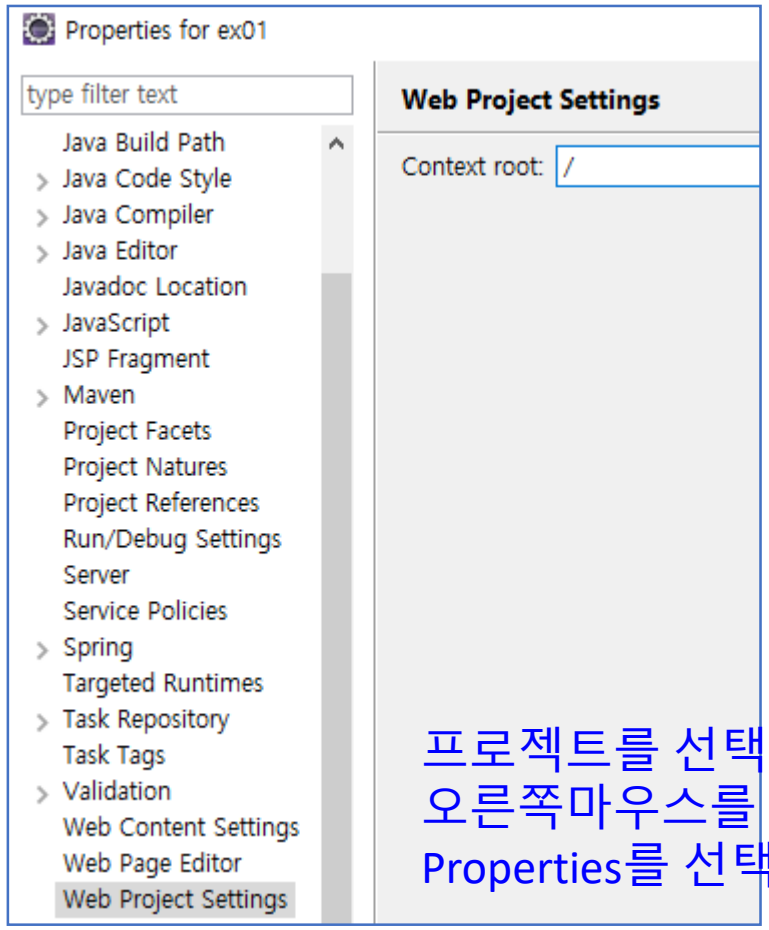
- 스프링 프로젝트의 버전 변경 => 5.2.7
- Java 버전 변경 => 14
- Junit 버전 => 4.12
- Log4j 버전 => 1.2.17
- Lombok 버전 => 1.18.24
- servlet 버전 => 4.0.1(이전 것과 groupId, artifactId 가 다르니 확인할 것)
- jsp 버전 => 2.3.3(이전 것과 groupId, artifactId 가 다르니 확인할 것)
- Mariadb => 2.7.5
- HikariCP => 3.4.5
- Mybatis => 3.5.6
- Mybatis-spring => 2.0.6
- Log4jdbc =>1.16

프로젝트의 경로(path)

- Tomcat을 이용하는 경로 변경



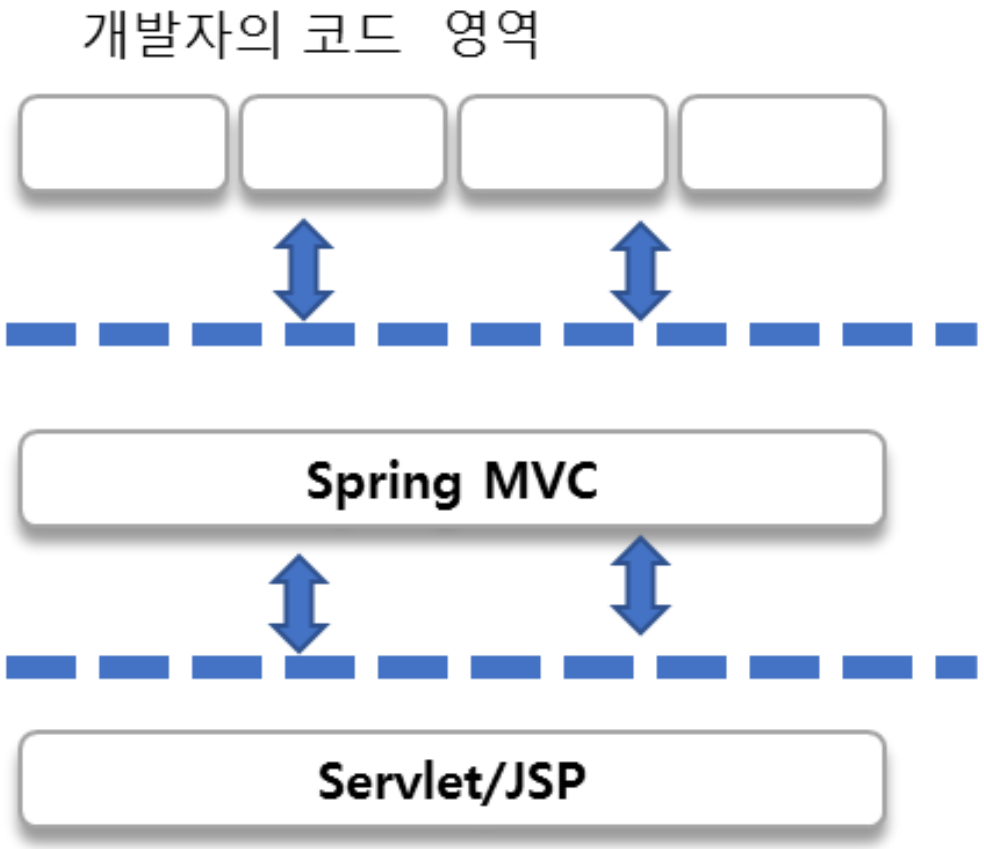
- Web Project Setting를 이용하는 변경



프로젝트를 선택하고
오른쪽마우스를 클릭하여
Properties를 선택

스프링 MVC의 기본 사상

- 서블릿 기반이긴 하지만 한 단계 더 추상화된 수준의 개발 지향
- 서블릿 API없이도 개발이 가능한 수준



기반 기술

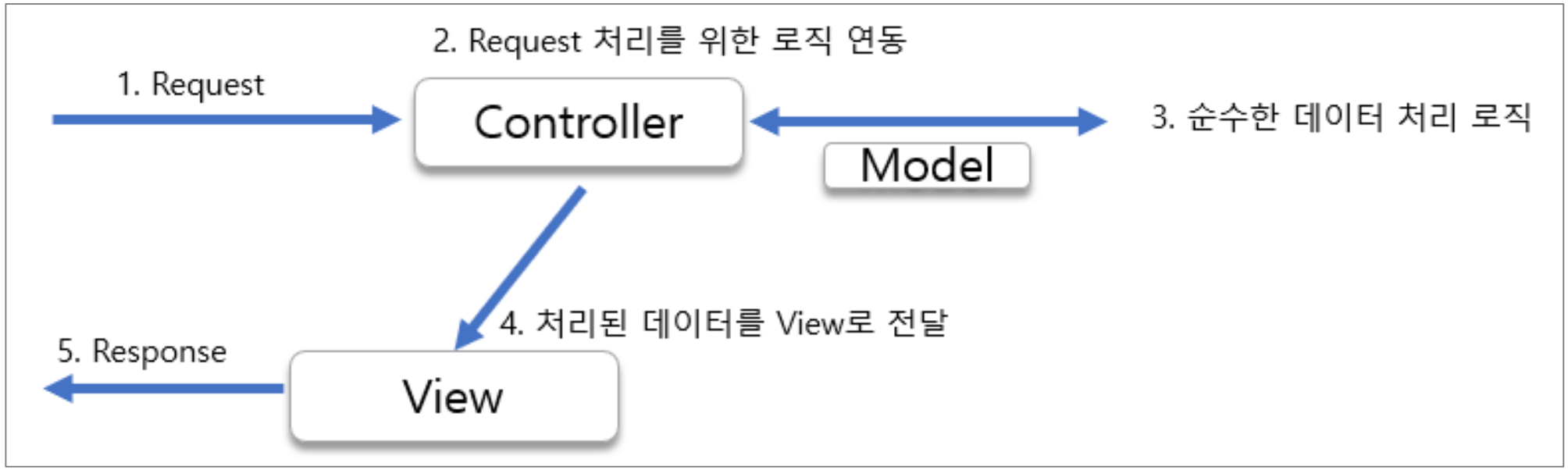
개발자는 Servlet/JSP의 API에 신경 쓰지 않고 웹 애플리케이션을 제작

Spring MVC는 내부적으로 Servlet/JSP 처리

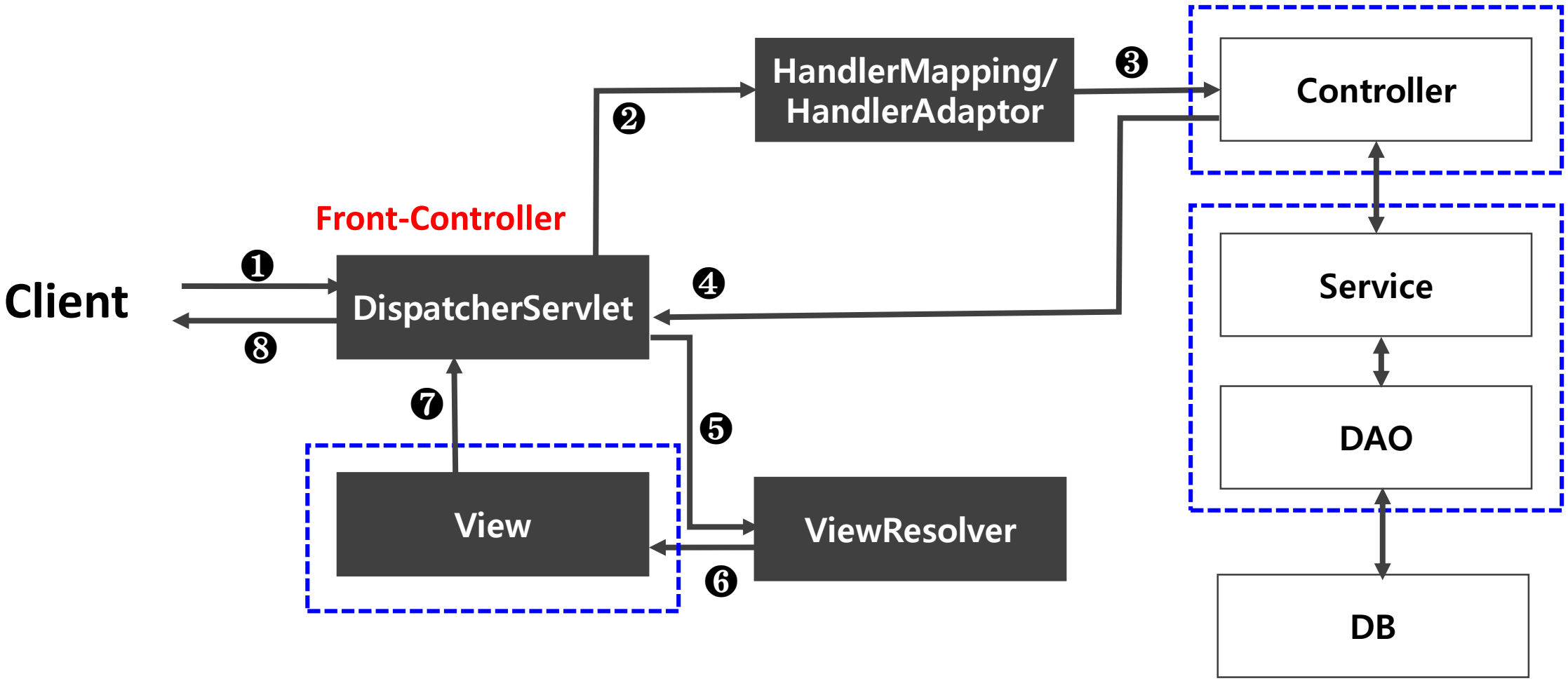
HttpServletRequest/ HttpServletResponse 객체를 사용할 필요성이 현저히 줄어듬

모델2 방식

- 화면과 로직을 분리하는 방식
  - 사용자 요청(Request)은 가정 먼저 Controller 를 호출
  - View를 교체하더라도 사용자가 호출하는 URL 자체에 변화가 없게 처리하기 위해



스프링 MVC의 기본 흐름(중요)



스프링 제공

개발자 구현

DispatcherServlet
내부적으로 스프링 컨테이너를 생성

스프링 MVC의 기본 흐름(중요)

❶ 사용자 요청(request)은 Front-Controller 인 **DispatcherServlet** 을 통해 처리됨
- 모든 request의 URL을 DispatcherServlet이 받도록 설정되어 있음(web.xml)

❷ **HandlerMapping**은 요청(request) 처리를 담당하는 컨트롤러를 찾기 위해 존재
- @RequestMapping 어노테이션이 적용된 것을 기준으로 판단

❸ 적절한 컨트롤러를 찾으면 **HandlerAdaptor**를 이용하여 해당 컨트롤러를 동작 시킴

❹ Controller는 개발자가 작성하는 클래스로 실제 요청(request)을 처리하는 로직 작성
- View에 전달해야 하는 데이터는 주로 **Model** 객체에 담아서 전달
- Controller는 다양한 타입의 결과를 반환하는데 이는 **ViewResolver**가 담당함

❺ **ViewResolver**는 Controller가 반환한 결과를 어떤 View를 통해서 처리하는 것이 좋을지 해석하는 역할
- Servlet-context.xml에 정의된 InternalResourceViewResolver 사용

❻ View는 실제로 응답보내야 하는 데이터를 jsp 등을 이용해 생성하는 역할

❼ 처리 결과가 포함된 View를 DispatcherServlet로 송신

❽ 최종결과가 클라이언트로 전달됨

스프링 MVC의 기본 흐름(중요)

- 프로젝트 구동 시 관여하는 설정 파일
  1. **web.xml** : tomcat 구동과 관련된 설정
     - 스프링 컨테이너 생성
     - root-context.xml 을 읽어들이도록 함
  2. **root-context.xml** : 스프링 객체 관련 설정
  3. **servlet-context.xml** : 스프링 웹 관련 설정

web.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee https://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">

    <!-- The definition of the Root Spring Container shared by all Servlets and Filters -->
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/spring/root-context.xml</param-value>
    </context-param>

    <!-- Creates the Spring Container shared by all Servlets and Filters -->
    <listener>
        <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
    </listener>

    <!-- Processes application requests -->
    <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>
</web-app>
```

FrontController 생성

servlet-context.xml

- InternalReourceViewResolver : 뷰를 처리하는 해결사

```
<!-- Resolves views selected for rendering by @Controllers to .jsp resources in the /WEB-IN
<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>
```

```
@Controller
public class HomeController {

    private static final Logger logger = LoggerFactory.getLogger(HomeController.class);

    @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, DateFor

        String formattedDate = dateFormat.format(date);

        model.addAttribute("serverTime", formattedDate );

        return "home";
    }
}
```

prefix

/WEB-INF/views/home.jsp 파일로 이동

prefix suffix