📚 Spring Study 📚
ᘏ3주차ᘏ

모두가 스프링 신이 되는 그날까지 불태우자~! 🔥🔥

“MVC패턴과 Spring 파일 구조”

IntelliJ의 프로젝트 구조

구분	Eclipse	IntelliJ
Workspace	Workspace	Project
Project	Project	Module

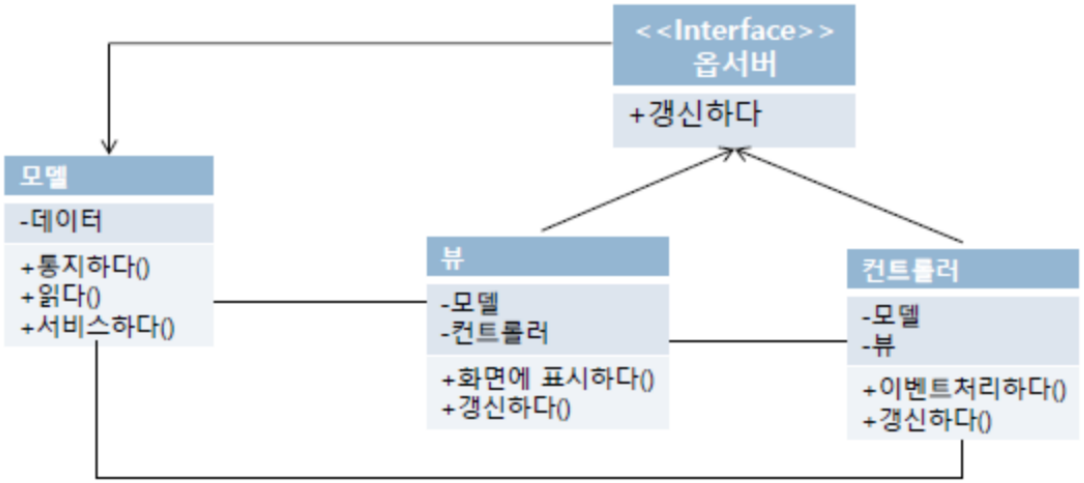
➡️ IntelliJ는 Eclipse와 달리 Workspace가 Project이고, Project가 Module이다

➡️ IntelliJ에서는 “한 프로젝트” 내에 “여러개의 모듈”로 구성되어 있다!

MVC 패턴

애플리케이션의 역할을 모델(Model), 뷰(View), 컨트롤러(Controller)로 나누어 작업

- 업무 서비스와 도메인 객체를 사용자 인터페이스로부터 분리, 하나 이상의 컨트롤러를 통해서 이들 사이의 상호작용을 통제하는 아키텍처 패턴



📚 MVC 모델 구성요소

✏️ 모델(Model) : 데이터를 실어서 보내준다.
애플리케이션의 핵심적인 기능 제공
뷰와 컨트롤러 등록
데이터 변경 시 뷰와 컨트롤러에 통지

✏️ 뷰(View)
관련된 컨트롤러 생성, 초기화, 사용자에게 정보 표시
데이터 변경이 통지될 때 모델로부터 데이터를 읽어 처리

✏️ 컨트롤러(Controller)
사용자 입력을 이벤트로 받아들여 해석하여 모델에 대한 요청을 서비스하거나 뷰에 대한 요청을 표시
데이터 변경이 통지될 때 모델로부터 데이터 읽어와 처리
Front Controller 패턴 적용 – 컨트롤러가 중심이 되는 변형된 구조를 가짐

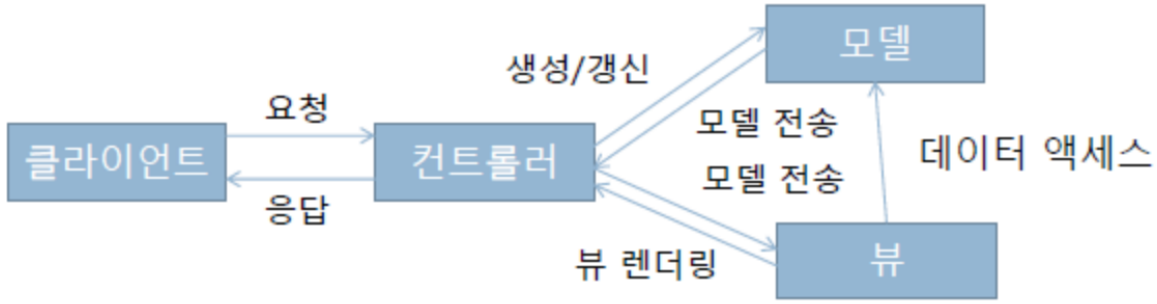
MVC 동작

클라이언트가 컨트롤러에게 데이터 조회, 저장 등의 작업 요청

- 컨트롤러 : 서블릿으로 구현, 업무 로직 처리, 모델생성, 모델의 데이터 갱신, 뷰에 모델 전송,
  뷰가 변경된 데이터 사용 가능하게 함

- 모델 : POJO Java 클래스로 구현, 데이터와 데이터처리에 필요한 메서드 포함, 자신이 관리하는 데이터에 집중

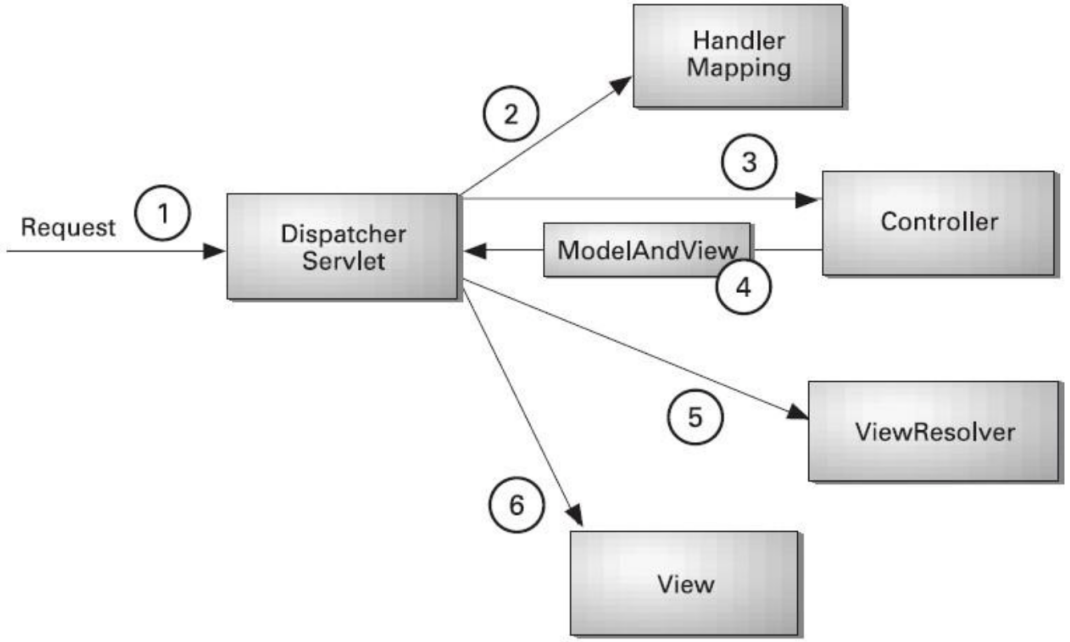
- 뷰 : JSP로 구현, 컨트롤러가 제공한 모델을 사용하여 데이터에 액세스하여 웹페이지(뷰)를 렌더링하여
  응답을 생성하고 컨트롤러에 전달 , 컨트롤러는 클라이언트에 응답을 전송하고 서비스 종료



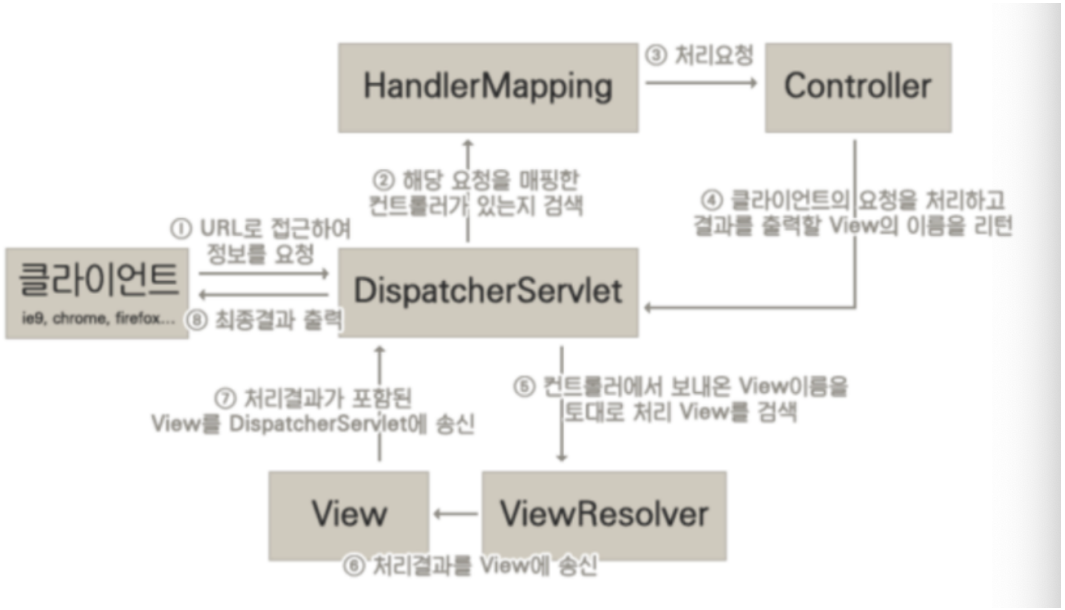
Spring MVC 매커니즘

기본으로 Servlet MVC 패턴은 Dispatcher Servlet을 제공 + annotation 제공

1. 핸들러 맵핑 : 어떤 요청이 들어왔는지 확인한다.
2. 실질적으로 일을 처리할 컨트롤러를 선택한다.
3. 처리 결과를 Model (데이터를 실어준다)과 논리 View(데이터를 표시)로 다시 Dispatcher Servelt으로 돌려준다.
4. ViewResolver, 진짜 물리적인 뷰를 찾아서 내용을 보여준다.

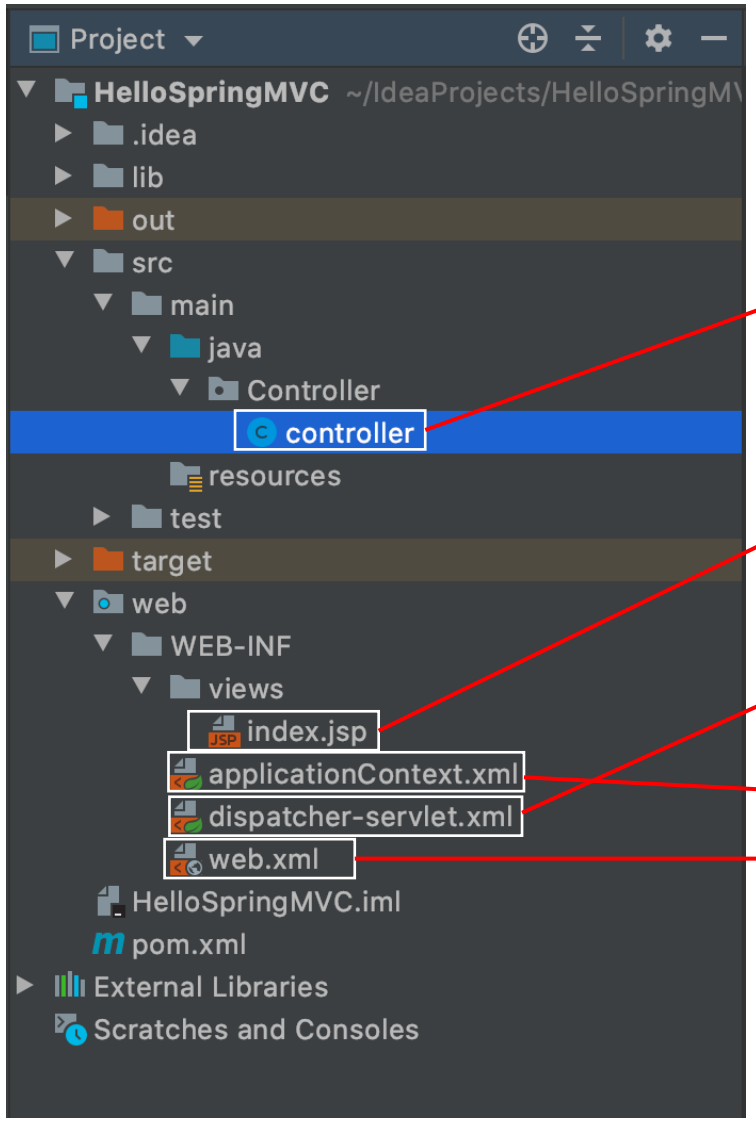


Spring MVC 매커니즘



클라이언트의 모든 요청은 (web.xml에 설정되어 있는 대로) 우선 DispatcherServlet이 받는다.
DispatcherServlet은 HandlerMapping을 통해 그 요청에 해당하는 컨트롤러가 존재하는지 확인한다.
요청에 해당하는 Controller가 있다면, 요청에 대한 처리를 위임해준다.
Controller는 그에 대한 처리를 끝내고 String으로 return을 하게 되는데, 이때 리턴한 String 문자열은 View의 이름이다.
다시 DispatcherServlet은 return되어 온 String을 토대로 ViewResolver(servlet-context.xml)를 통해 동일한 이름의 View가 있는지 검색한 후 동일한 이름의 View가 있다면 응답, 요청 객체를 그 view에 송신을 하고
그 View에서 그에 대한 처리를 한 다음 다시 DispatcherServlet에 보내면 DispatcherServlet이
html로 해석해서 다시 클라이언트에게 보내준다.

Spring MVC



🛠️ 지난 프로젝트를 살펴보면 🛠️

컨트롤러
: Dispatcher에서 전달된 요청을 처리

뷰(.jsp)

DispatcherServlet
1) 클라이언트의 요청을 최초 받아
2) 컨트롤러에게 전달

web.xml
1) Dispatcher Servlet 서블릿 맵핑
2) 스프링 설정 파일 위치 정의

applicationContext.xml
: 스프링 컨테이너 설정 파일

Spring 디렉토리 구조 살펴보기

- src/ main/ java – 자바 클래스 (controller, DAO, VO 등)
- src/ main/ resources – **자바 클래스들이 사용될 각종 맵퍼 (config.xml,mapper 등) 와 설정 파일**
- src/ main/ test – 테스트 영역
- web/WEB-INF/views – **view, 말 그대로 jsp.html영역**
- web/WEB-INF/web.xml – **배포서술자(Deployment Descriptor) 라고 하며, 해당 파일 내에 정의된 설정내용 구성**
- pom.xml – **Dependency를 설정해줌으로써 , 의존성관리와 배포를 가능하게 해줌**
  => 메이븐 설정파일로 메이븐은 라이브러리를 연결해주고, 빌드를 위한 플랫폼

프로젝트 내 Spring 파일(디렉토리) 구조

➡️ web.xml

Servelt 지정

```xml
<?xml version="1.0" encoding="UTF-8"?>
<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">
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/applicationContext.xml</param-value>
    </context-param>
    <listener>
        <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
    </listener>
    <servlet>
        <servlet-name>dispatcher</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
    </servlet>
```

Servelt 매핑

```xml
    <servlet-mapping>
        <servlet-name>dispatcher</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>
```

프론트 컨트롤러 DispatcherServlet

** DispatcherServlet

- HTTPServlet 클래스에서 파생된 서블릿

- 매핑된 URL의 모든 HTTP 요청을 처리

- 각 DispatcherServlet은 자기 자신의 웹 어플리케이션 용 IoC 컨테이너인 WebApplicationContext가 생성

- WebApplicationContext는 Spring MVC 웹 어플리케이션에 필요한 Spring 빈의 인스턴스를 생성, 관리

- 생성된 Spring빈은 웹 에플리케이션에서 정의한 컨트롤러, HTTP 요청을 컨트롤러와 매핑시켜주는 HandlerMapping과 ViewResolver도 포함한다.

개발환경 내 Spring 디렉토리 구조

디렉토리 및 파일								설명
업무 프로젝트 (ex: display-api)	src	main	java	com.cname (도메인)	서비스 (ex: display)	업무 영역 api fo bo	common	프로젝트 내에서 사용하는 공통 모듈
							controller	서블릿 요청을 처리하는 컨트롤러
							service	비지니스 로직 처리
							dao	데이터 access를 위한 로직
							model	데이터를 저장하는 POJO 객체
							interceptor	controller 클래스 호출 전에 비지니스 로직을 처리
						config		프로젝트 내의 configuration 설정 Bean
						exception		전역 예외처리 클래스
			resources	application.ym				업무 프로젝트의 환경별 설정 및 property 정보 저장
				mapper				myvatis 관련 설정 및 xml query 파일 저장
				logback.xml				업무프로젝트별 로그 설정 저장
				static	js			업무 로직 javascript 파일 (FO/MO만 해당)
				templates	서비스			Thymeleaf 파일 저장
		test	java					테스트에 필요한 Junit 테스트 케이스
			resources					테스트에 필요한 설정 관련 파일
	.project							STS 프로젝트 설정 파일
	.classpath							STS 클래스패스 설정 파일
	.gitignore							git 제외대상 파일 등록
	pom.xml							Maven 설정 파일