10

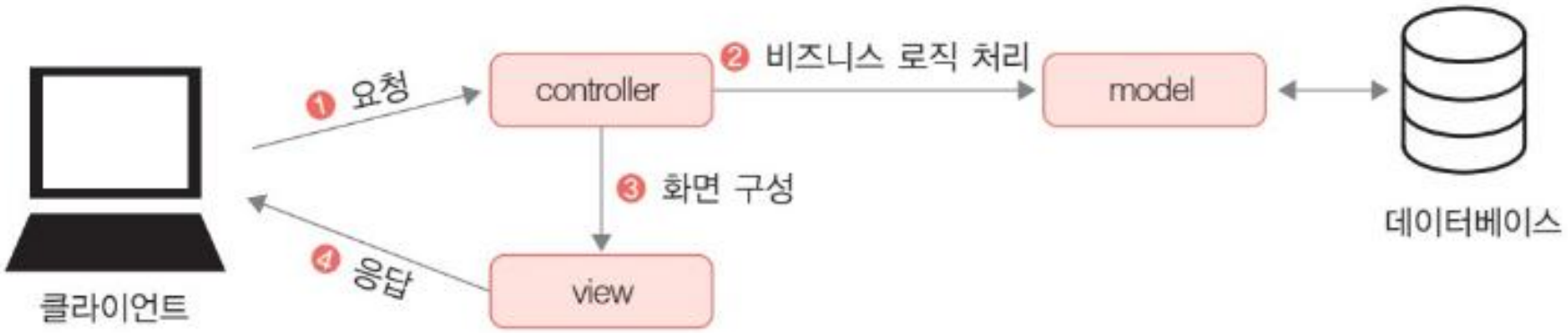
Spring Boot에서 JSP 구현하기

2

MVC 기본 개념

MVC 패턴 이해하기

- **MVC 개념**
  - **Model, View, Controller**를 이용해서 프로그래밍하는 설계 방법
  - 구성 요소
    - **Model** - **DB와 밀접한 관계**를 갖고 **비즈니스 로직**을 담당
    - **View** - **클라이언트와 밀접한 관계**를 갖고 비즈니스 로직의 결과를 출력하기 위한 **화면 구성**을 담당
    - **Controller** – **클라이언트의 요청**에 대해 Model과 View를 컨트롤하는 업무를 담당

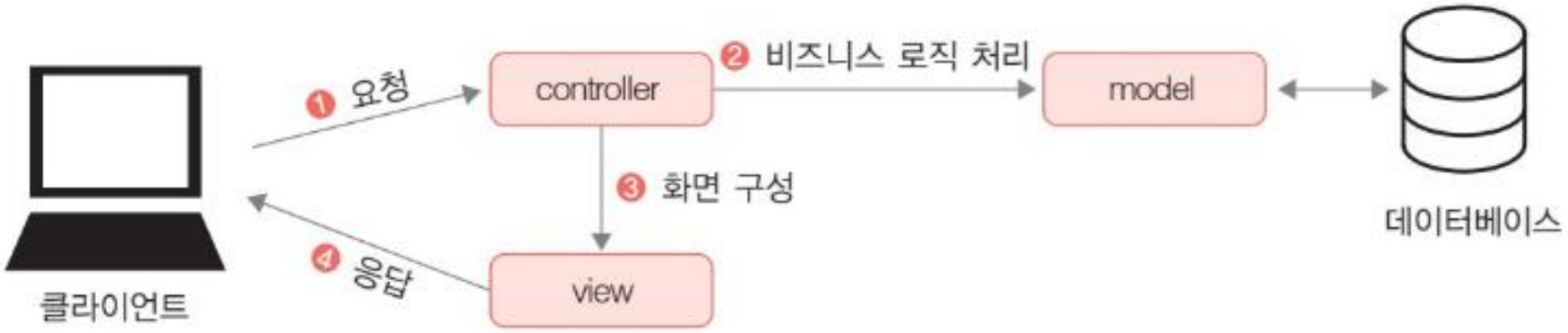


각각 업무를 분업화 하는 MVC 패턴

MVC 패턴 이해하기

- 스프링 부트 프로젝트 => 웹 프로젝트
  - 웹 **브라우저 주소창**에 **URL을 입력**하면(**요청** 단계)
  - **요청 형태에 따라 메서드 호출(로직 처리** 단계)
  - **로직 처리 결과를 보여줄 페이지(화면구성** 단계)가 호출되는 방식 (**응답** 단계)

  ➡ **@Controller 어노테이션**을 적용한 클래스에서 처리



각각 업무를 분업화 하는 MVC 패턴

5

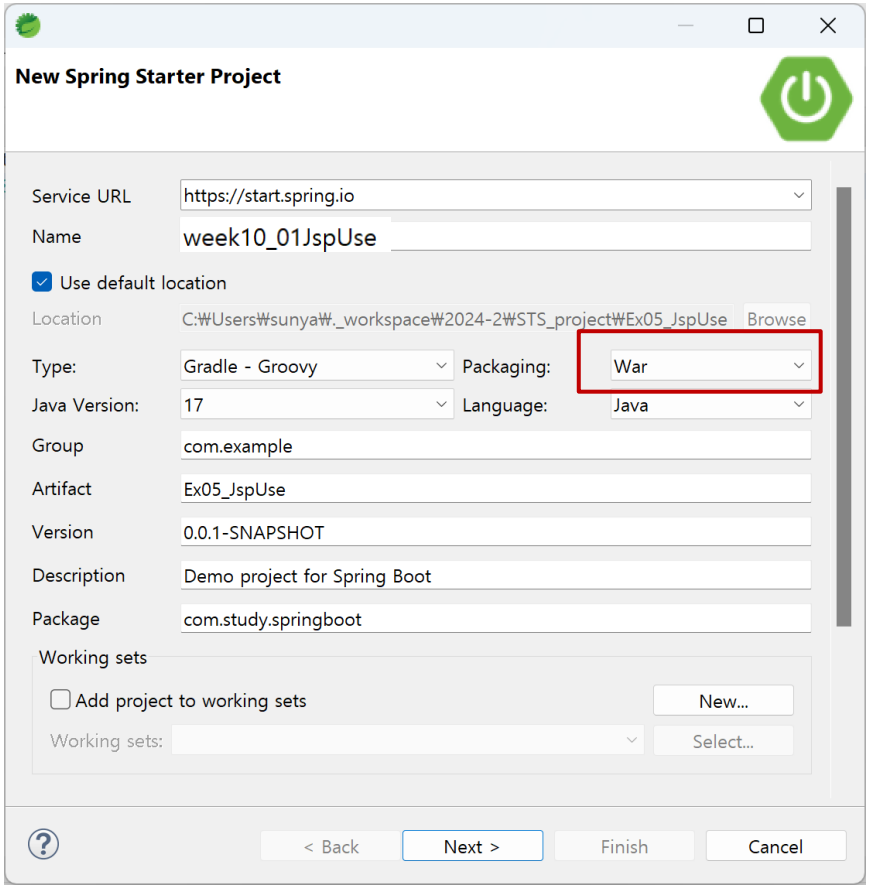
Spring Boot에서 JSP 구현하기

JSP 사용하기

- **스프링 부트**에서 **JSP 사용**
  - 스프링 부트에서 JSP를 사용하기 위해서는 **추가적인 설정**이 **필요**
  - **프로젝트 생성** 시 선택한 **war 타입**은 실행 가능한 war 파일로 만들었을 때 내장 WAS로 실행하거나 외부 WAS에 배포해도 JSP가 정상 작동함

JSP 사용하기

- JSP 사용을 위한 프로젝트 기본 설정
  - 스프링 부트 프로젝트 생성 > **week10_01JspUse**
  - **SpringWeb** 의존성 추가



JSP 사용하기

- JSP 사용을 위한 프로젝트 기본 설정
  - 스프링 부트 프로젝트 생성 > **week10_01JspUse**
  - **build.gradle** 파일 > **JSP 사용을 위한 의존성 추가**

build.gradle

```
repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-web'
    providedRuntime 'org.springframework.boot:spring-boot-starter-tomcat'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    testRuntimeOnly 'org.junit.platform:junit-platform-launcher'
    implementation 'org.apache.tomcat.embed:tomcat-embed-jasper'
    implementation 'jakarta.servlet.jsp.jstl:jakarta.servlet.jsp.jstl-api'
    implementation 'org.glassfish.web:jakarta.servlet.jsp.jstl'
}

tasks.named('test') {
    useJUnitPlatform()
}
```

JSP 사용하기

프로젝트 생성 시 추가하지 못했거나, 프로젝트 개발 중간에 더 필요한 디펜던시가 생기면 아래와 같은 방법으로 build.gradle에 디펜던시를 추가할 수 있다.

- JSP 사용을 위한 프로젝트 기본 설정
  - 스프링 부트 프로젝트 생성 > **week10_01JspUse**
  - **build.gradle** 파일 > **JSP 사용을 위한 의존성 추가**

build.gradle

```
repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-web'
    providedRuntime 'org.springframework.boot:spring-boot-starter-tomcat'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    testRuntimeOnly 'org.junit.platform:junit-platform-launcher'
    implementation 'org.apache.tomcat.embed:tomcat-embed-jasper'
    implementation 'jakarta.servlet.jsp.jstl:jakarta.servlet.jsp.jstl-api'
    implementation 'org.glassfish.web:jakarta.servlet.jsp.jstl'
}

tasks.named('test') {
    useJUnitPlatform()
}
```

JSP 사용하기

*REST 기반 아키텍처를 사용하여 대규모의 고성능 통신을 안정적으로 지원 가능
*REST 아키텍처 스타일을 따르는 API => **RESTful API**

- JSP 사용을 위한 프로젝트 기본 설정
  - 스프링 부트 프로젝트 생성 > **week10_01JspUse**
  - **build.gradle** 파일 > **JSP 사용을 위한 의존성 추가**

build.gradle

```
repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-web'
    providedRuntime 'org.springframework.boot:spring-boot-starter-tomcat'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    testRuntimeOnly 'org.junit.platform:junit-platform-launcher'
    implementation 'org.apache.tomcat.embed:tomcat-embed-jasper'
    implementation 'jakarta.servlet.jsp.jstl:jakarta.servlet.jsp.jstl-api'
    implementation 'org.glassfish.web:jakarta.servlet.jsp.jstl'
}

tasks.named('test') {
    useJUnitPlatform()
}
```

Spring MVC를 사용해서 RESTful* 웹 서비스를 개발할 때 필요한 의존성 모음

JSP 사용하기

- JSP 사용을 위한 프로젝트 기본 설정
  - 스프링 부트 프로젝트 생성 > **week10_01JspUse**
  - **build.gradle** 파일 > **JSP 사용을 위한 의존성 추가**

build.gradle

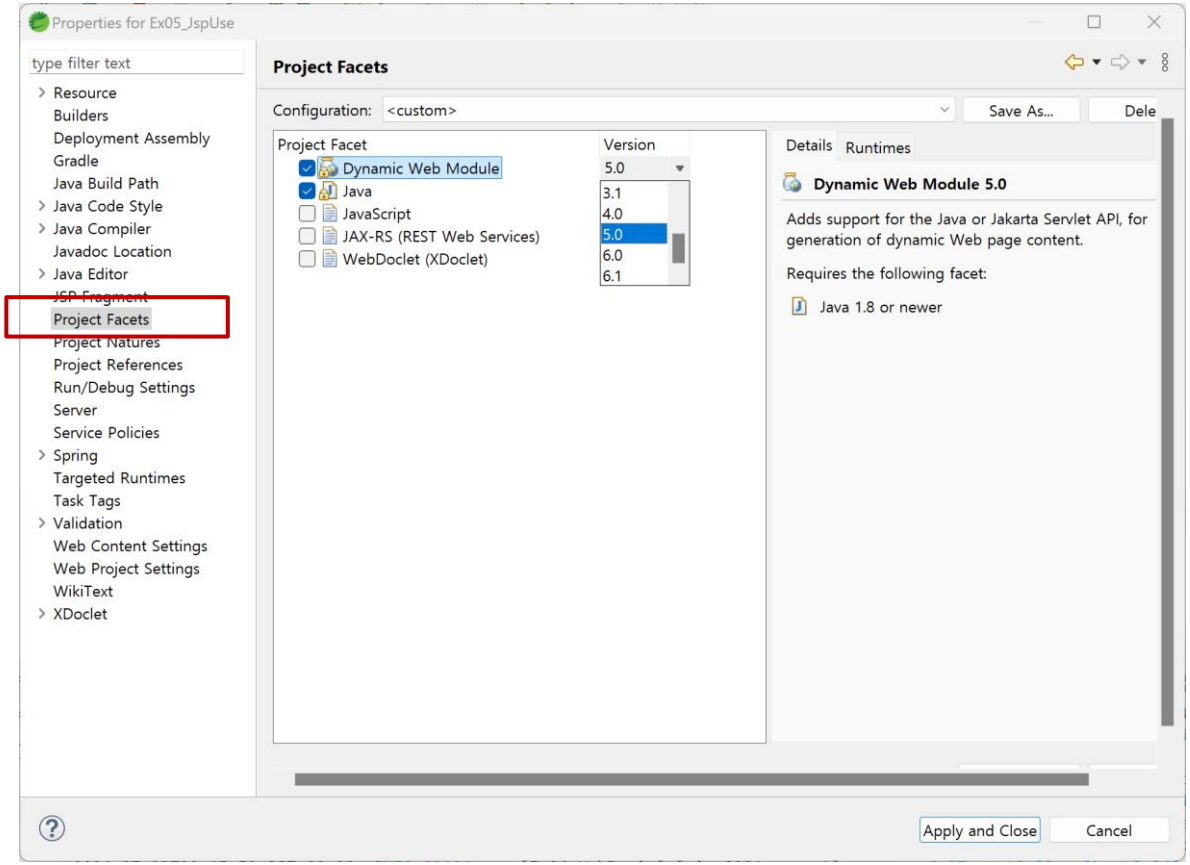
```
repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-web'
    providedRuntime 'org.springframework.boot:spring-boot-starter-tomcat'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    testRuntimeOnly 'org.junit.platform:junit-platform-launcher'
    implementation 'org.apache.tomcat.embed:tomcat-embed-jasper'
    implementation 'jakarta.servlet.jsp.jstl:jakarta.servlet.jsp.jstl-api'
    implementation 'org.glassfish.web:jakarta.servlet.jsp.jstl'
}

tasks.named('test') {
    useJUnitPlatform()
}
```

프로젝트명 or bulid.gradle 선택 – 우클릭 – 팝업메뉴 – Gradle – Refresh Gradle Project 실행

JSP 사용하기

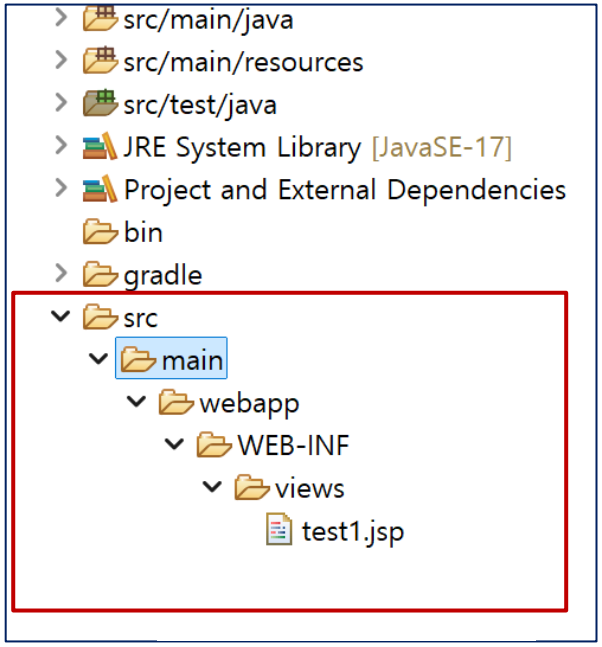
반드시 JSP 파일 생성 전에 실행

- JSP 사용을 위한 폴더 생성 전 설정
  - **프로젝트명** > 우클릭 > **Properties** > **Project Facets**
  - **Dynamic Web Module 버전** 수정 > **5.0**



JSP 사용하기

- **JSP 사용**을 위한 **폴더 생성**
  - 스프링 부트에서 기본으로 제공하는 다른 템플릿 뷰와는 달리 JSP는 src/main/resources의 템플릿 폴더를 사용할 수 없다.
  - JSP를 위해 필요한 폴더는 **직접 만들고 지정**



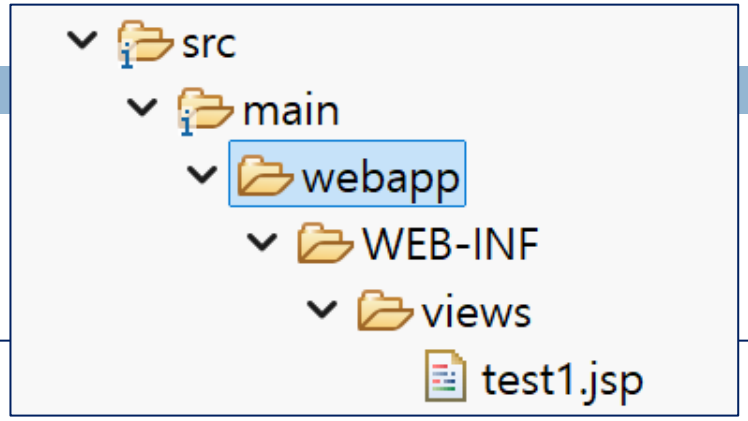
JSP 사용하기

- JSP 뷰 만들기
  - **test1.jsp**

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <h1>Hello JSP in SpringBoot~!</h1>
</body>
</html>
```

JSP 사용하기

- Controller 만들기
  - **MyController.java**

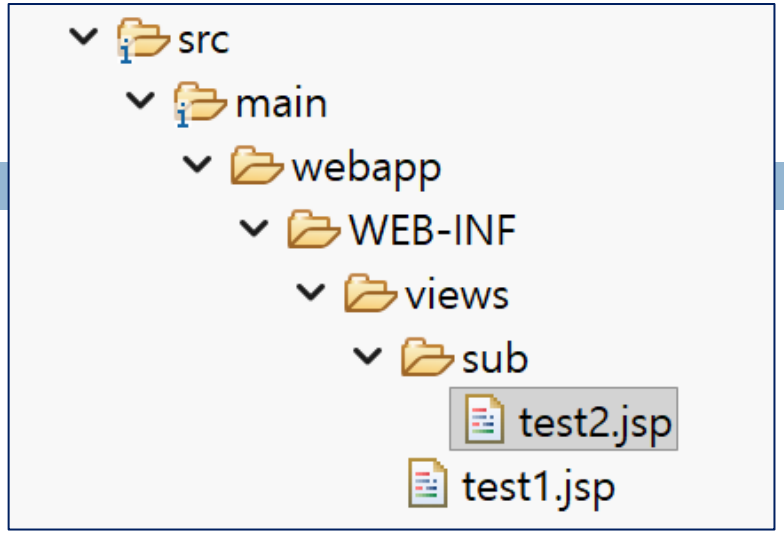



```
@Controller
public class MyController {
    @RequestMapping("/")
    @ResponseBody
    public String root() {
        return "SpringBoot에서 JSP 구현하기";
    }

    @RequestMapping("/test1")
    public String test1() {
        return "/WEB-INF/views/test1.jsp";
    }
}
```

webapp 폴더 - JSP의 root("/")

JSP 사용하기

- JSP 뷰 만들기
  - **test2.jsp**



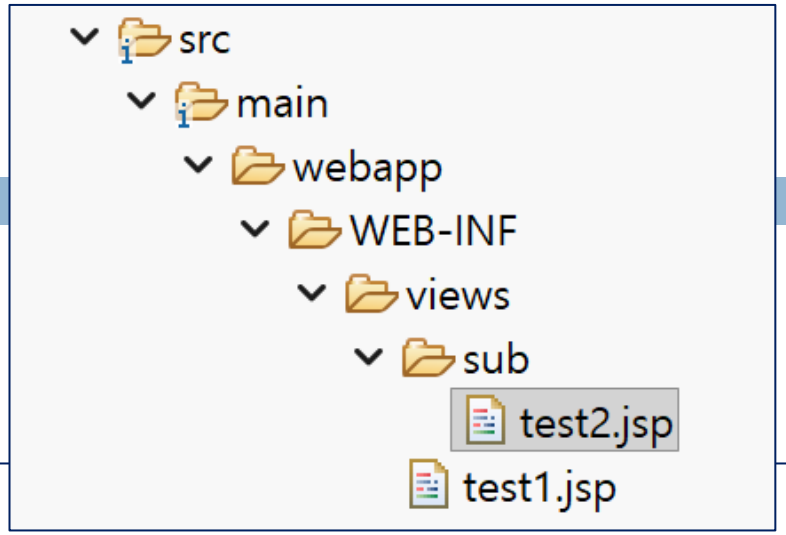

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <h1>Hello test2.jsp ~!</h1>
</body>
</html>
```

JSP 사용하기

- Controller 만들기
  - **MyController.java**

```
@Controller
public class MyController {
    @RequestMapping("/")
    @ResponseBody
    public String root() {
        return "SpringBoot에서 JSP 구현하기";
    }

    @RequestMapping("/test1")
    public String test1() {
        return "/WEB-INF/views/test1.jsp";
    }

    @RequestMapping("/test2")
    public String test2() {
        return "/WEB-INF/views/sub/test2.jsp";
    }
}
```

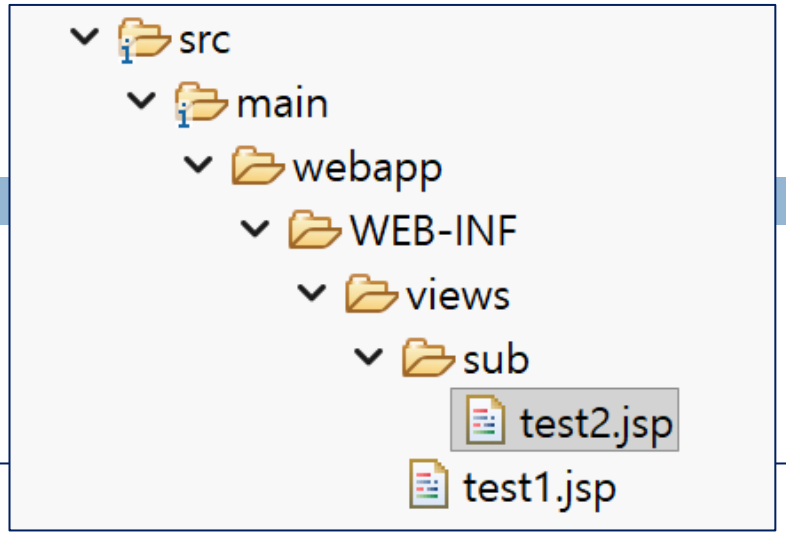
JSP 사용하기



- Controller 만들기
  - **MyController.java**

```
    @RequestMapping("/test1")
    public String test1() {
        return "/WEB-INF/views/test1.jsp";
    }

    @RequestMapping("/test2")
    public String test2() {
        return "/WEB-INF/views/sub/test2.jsp";
    }
}
```

JSP 사용하기



- Controller 만들기
  - **MyController.java**

```
    @RequestMapping("/test1")
    public String test1() {
        return "/WEB-INF/views/test1.jsp";
    }

    @RequestMapping("/test2")
    public String test2() {
        return "/WEB-INF/views/sub/test2.jsp";
    }
}
```

JSP 사용하기

- JSP 사용이 가능하도록 설정
  - src/main/resources > **application.properties** 파일

application.properties

```
1 spring.application.name=week10_01JspUse
2 spring.mvc.view.prefix=/WEB-INF/views/
3 spring.mvc.view.suffix=.jsp
4
```

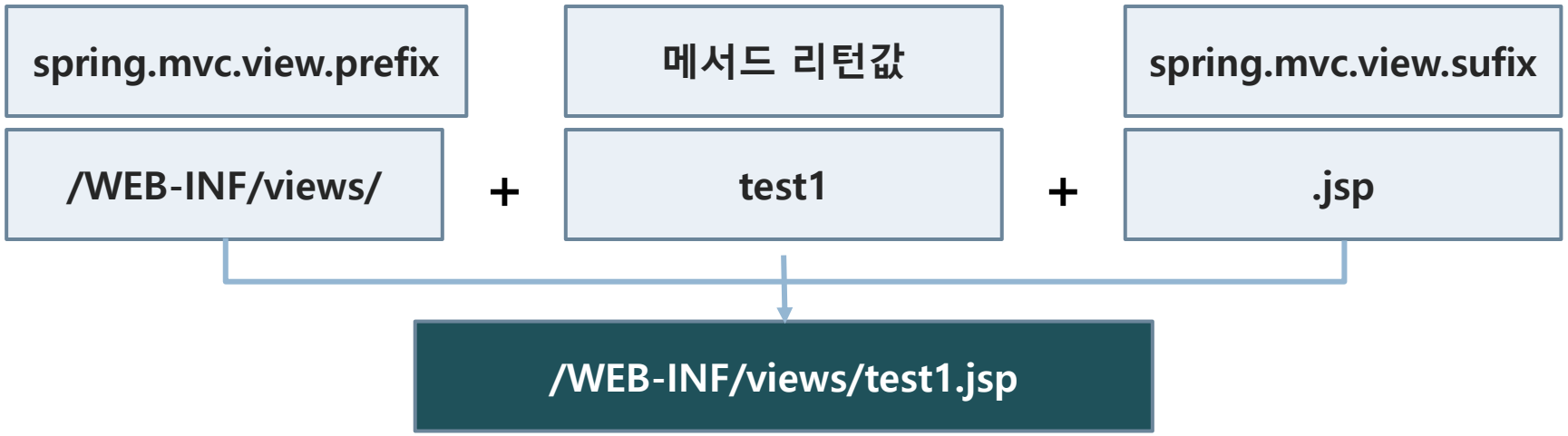
JSP 사용하기

application.properties

```
spring.application.name=week10_01JspUse
spring.mvc.view.prefix=/WEB-INF/views/
spring.mvc.view.suffix=.jsp
```

- **RequestMapping과 리턴값**
  - **@ResponseBody 어노테이션이 없는 경우**

```
@Controller
public class MyController {
    @RequestMapping("/")          // localhost:8081/ 호출 시 실행
    @ResponseBody                 // 리턴값을 직접 화면에 출력
    public String root() throws Exception {
        return "JSP in Gradle";
    }

    @RequestMapping("/test1")     // localhost:8081/test1 호출 시 실행
    public String test1() {
        return "test1";           // 실제 호출 될 /WEB-INF/views/test1.jsp
    }
```



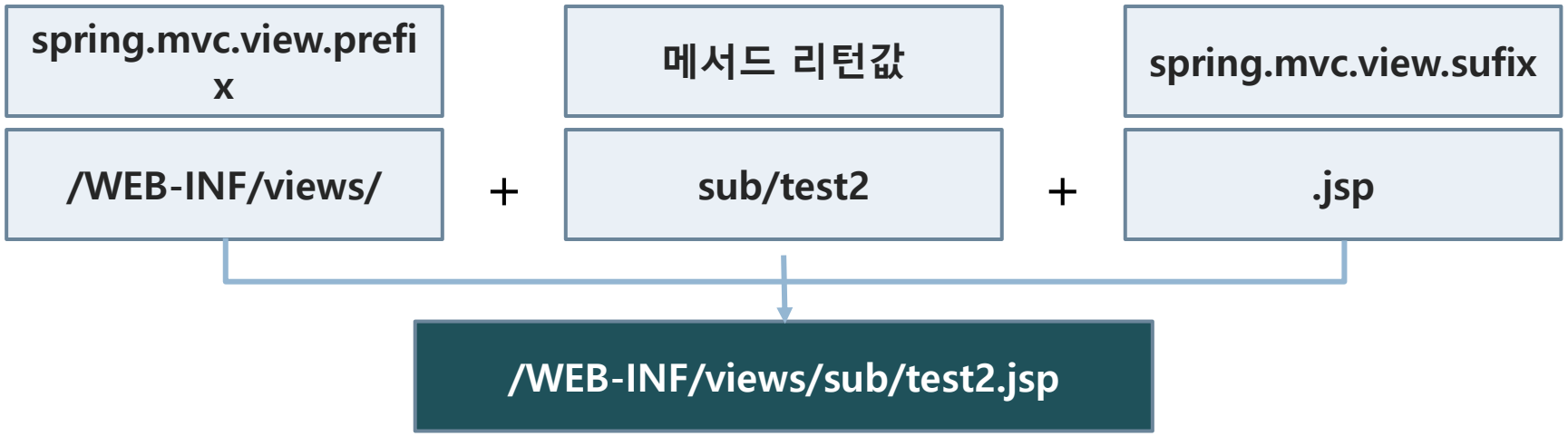
JSP 사용하기

application.properties

```
spring.application.name=week10_01JspUse
spring.mvc.view.prefix=/WEB-INF/views/
spring.mvc.view.suffix=.jsp
```

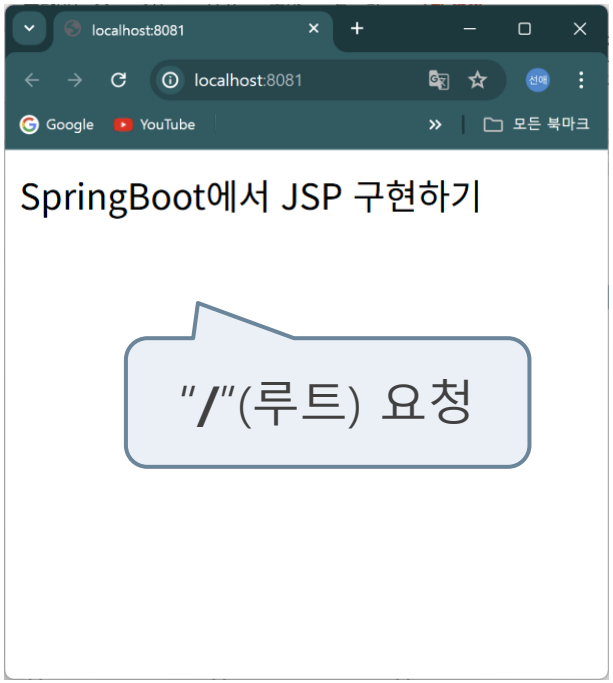
- **RequestMapping과 리턴값**
  - **@ResponseBody 어노테이션이 없는 경우**

```
14
15    @RequestMapping("/test1")    // localhost:8081/test1 호출 시 실행
16    public String test1() {
17        return "test1";          // 실제 호출 될 /WEB-INF/views/test1.jsp
18    }
19
20    @RequestMapping("/test2")    // localhost:8081/test2 호출 시 실행
21    public String test2() {
22        return "sub/test2";      // 실제 호출 될 /WEB-INF/views/sub/test2.
23    }
```



JSP 사용하기

- 프로젝트 실행

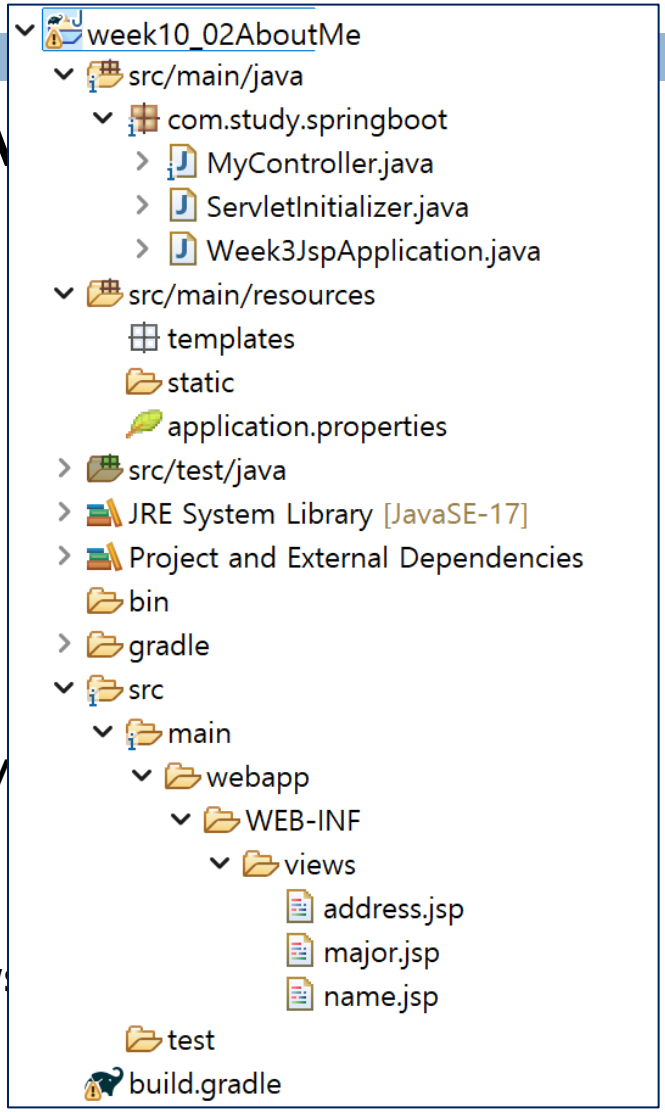


스프링 부트에서 JSP 사용 실습

- 프로젝트 생성 > **week10_02AboutMe**
- 프로젝트 구성
  - **build.gradle**에 **디펜던시 추가**
  - **Gradle** > **Refresh Gradle Project**
  - 프로젝트 > properties > Project Facets
    - **Dynamic Web Module > 5.0**
  - **resources** 폴더
    - **application.properties** 수정
      - **spring.mvc.view.prefix=/WEB-INF/views/**
      - **spring.mvc.view.suffix=.jsp**
    - **JSP 폴더 생성**
      - src > main > **webapp > WEB-INF > views**
    - jsp 파일 생성
      - **views 폴더에 생성**

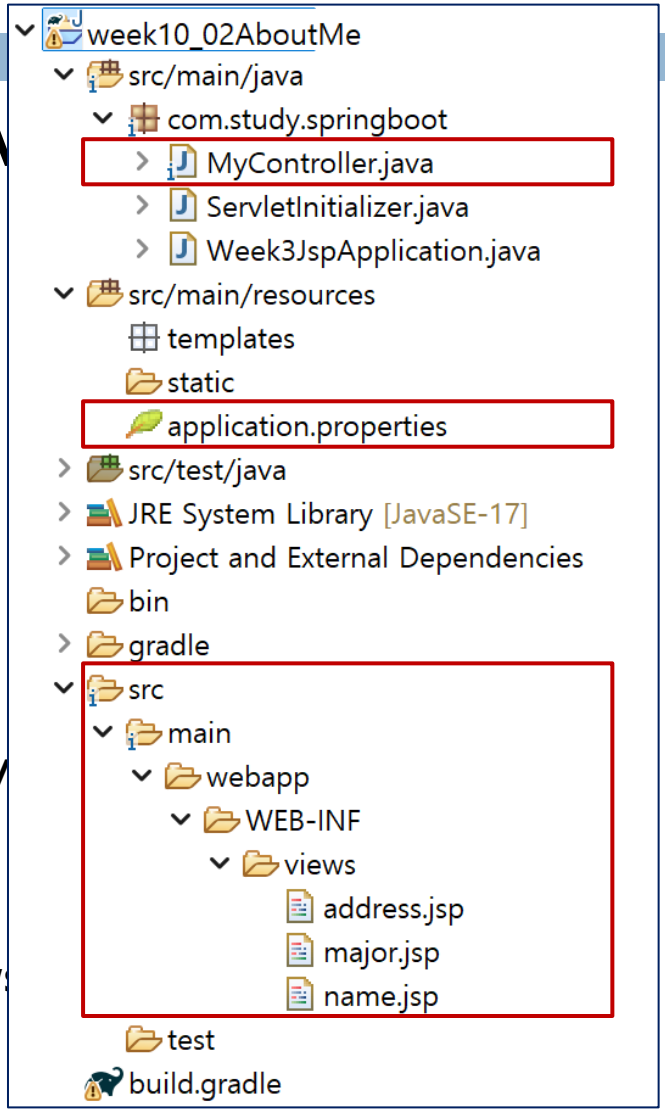
스프링 부트에서 JSP 사용 실습

- 프로젝트 생성 > **week10_02AboutM**
- 프로젝트 구성
  - **build.gradle**에 **디펜던시 추가**
  - **Gradle** > **Refresh Gradle Project**
  - 프로젝트 > properties > Project Facets
    - **Dynamic Web Module > 5.0**
  - **resources** 폴더
    - **application.properties** 수정
      - **spring.mvc.view.prefix=/WEB-INF/views/**
      - **spring.mvc.view.suffix=.jsp**
    - **JSP 폴더 생성**
      - src > main > **webapp > WEB-INF > views**
    - jsp 파일 생성
      - **views 폴더에 생성**



스프링 부트에서 JSP 사용 실습

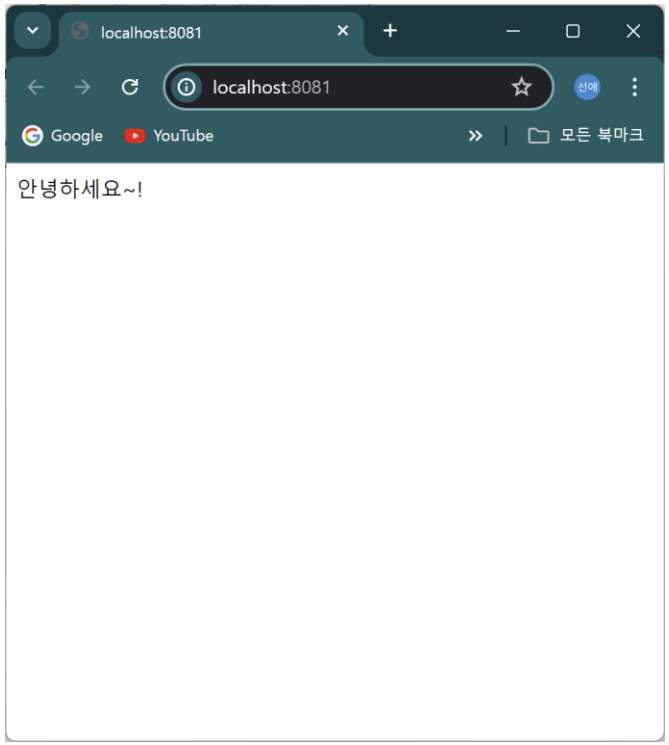
- 프로젝트 생성 > **week10_02AboutM**
- 프로젝트 구성
  - **build.gradle**에 **디펜던시 추가**
  - **Gradle** > **Refresh Gradle Project**
  - 프로젝트 > properties > Project Facets
    - **Dynamic Web Module > 5.0**
  - **resources** 폴더
    - **application.properties** 수정
      - **spring.mvc.view.prefix=/WEB-INF/views/**
      - **spring.mvc.view.suffix=.jsp**
    - **JSP 폴더 생성**
      - src > main > **webapp > WEB-INF > views**
    - jsp 파일 생성
      - **views 폴더에 생성**



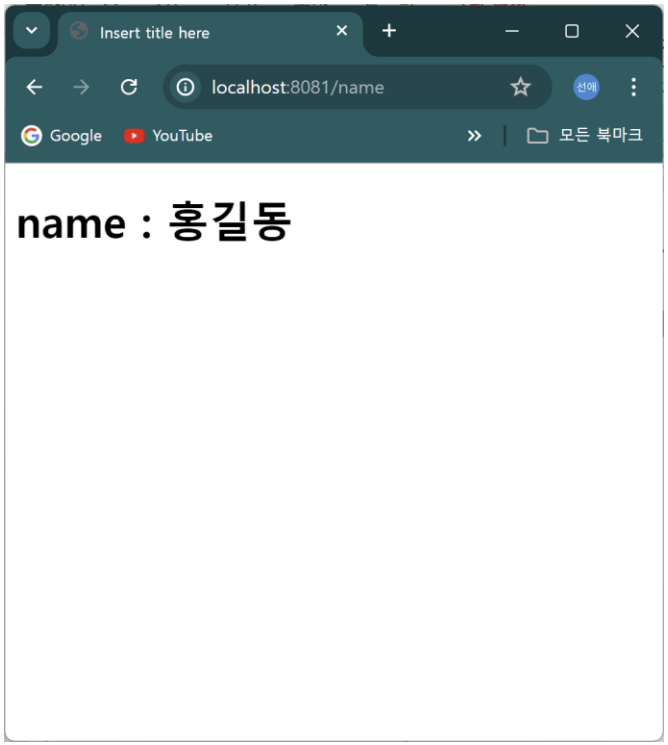
스프링 부트에서 JSP 사용 실습

- 프로젝트 생성 > **week10_02AboutMe**
- 요청 및 응답 관계

`@RequestMapping("/")`



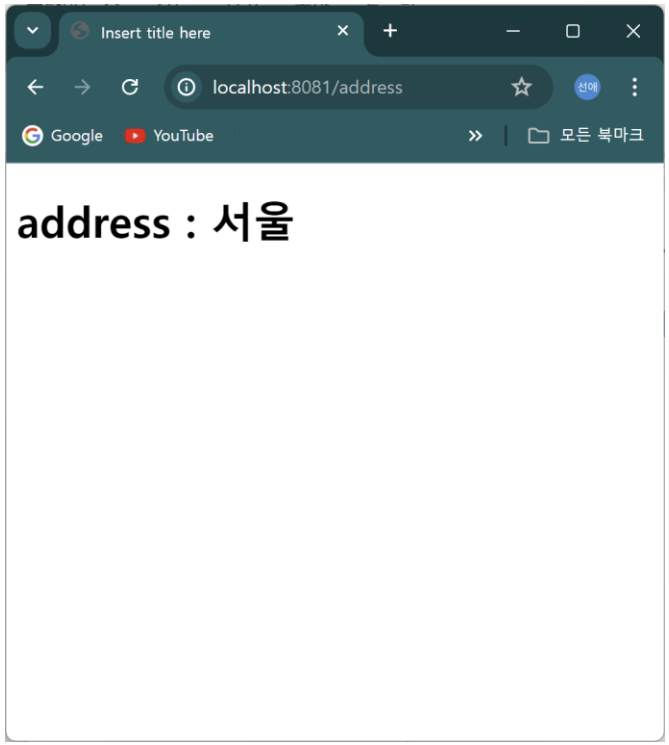
`@RequestMapping("/name")`



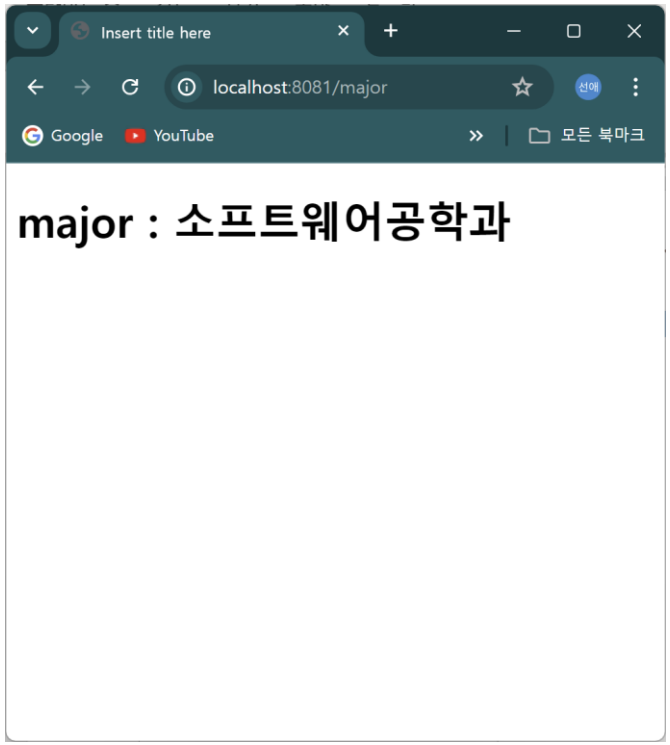
스프링 부트에서 JSP 사용 실습

- 프로젝트 생성 > **week10_02AboutMe**
- 요청 및 응답 관계

`@RequestMapping("/address")`



`@RequestMapping("/major")`



스프링 부트에서 JSP 사용 실습

- build.gradle

build.gradle

```
repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-web'
    providedRuntime 'org.springframework.boot:spring-boot-starter-tomcat'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
    testRuntimeOnly 'org.junit.platform:junit-platform-launcher'
    implementation 'org.apache.tomcat.embed:tomcat-embed-jasper'
    implementation 'jakarta.servlet.jsp.jstl:jakarta.servlet.jsp.jstl-api'
    implementation 'org.glassfish.web:jakarta.servlet.jsp.jstl'
}

tasks.named('test') {
    useJUnitPlatform()
}
```

스프링 부트에서 JSP 사용 실습

- jsp files

name.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html><html><head><meta charset="UTF-8">
<title>Insert title here</title></head>
<body>

                    <h1>name : 홍길동</h1>

</body></html>
```

address.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html><html><head><meta charset="UTF-8">
<title>Insert title here</title></head>
<body>

                    <h1>address : 경기도 화성시</h1>

</body></html>
```

major.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html><html><head><meta charset="UTF-8">
<title>Insert title here</title></head>
<body>

                    <h1>major : 컴퓨터공학과</h1>

</body></html>
```

스프링 부트에서 JSP 사용 실습

- MyController

application.properties

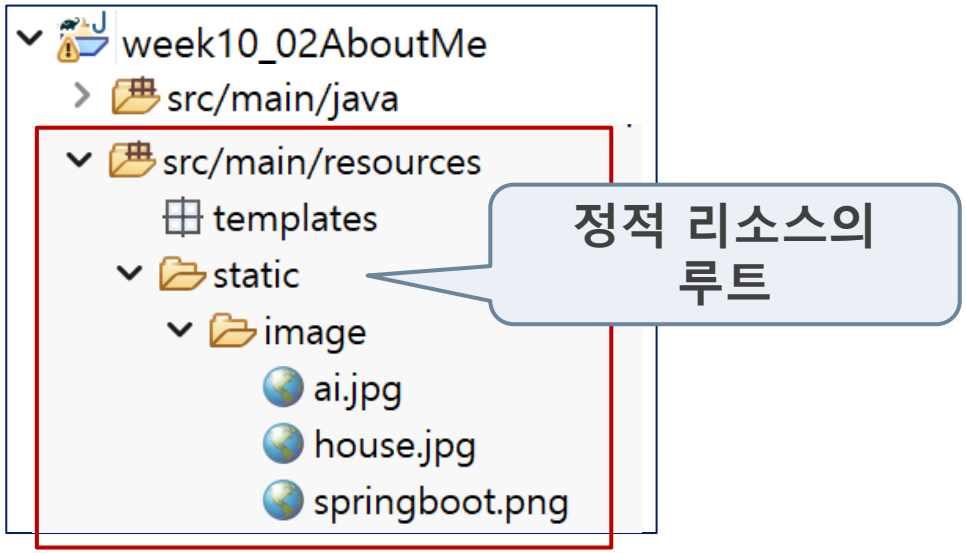
```
1
2
3 spring.mvc.view.prefix=/WEB-INF/views/
4 spring.mvc.view.suffix=.jsp
5
```

MyController.java

```
1  package com.study.springboot;
2
3  import org.springframework.stereotype.Controller;
6
7  @Controller
8  public class MyController {
9      @RequestMapping("/")          //localhost:8081/ 호출 시 실행
10     @ResponseBody                 //리턴값을 직접 화면에 출력
11     public String root() throws Exception {
12         return "안녕하세요~!";
13     }
14
15     @RequestMapping("/name")
16     public String method1() {
17         return "name";
18     }
19
20     @RequestMapping("/address")
21     public String method2() {
22         return "address";
23     }
24
25     @RequestMapping("/major")
26     public String method3() {
27         return "major";
28     }
29 }
```

스프링 부트에서 정적 리소스 사용하기

- 정적 리소스 - 이미지



스프링 부트에서 정적 리소스 사용하기

- jsp files

name.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html><html><head><meta charset="UTF-8">
<title>Insert title here</title></head>
<body>

                    <h1>name : 홍길동</h1>
                    <img src="/image/springboot.png">
</body></html>
```

address.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html><html><head><meta charset="UTF-8">
<title>Insert title here</title></head>
<body>

                    <h1>address : 경기도 화성시</h1>
                    <img src="/image/house.jpg">
</body></html>
```

major.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html><html><head><meta charset="UTF-8">
<title>Insert title here</title></head>
<body>

                    <h1>major : 컴퓨터공학과</h1>
                    <img src="/image/ai.jpg">
</body></html>
```

스프링 부트에서 정적 리소스 사용하기

- Controller

```
@Controller
public class MyController {

    @RequestMapping("/")
    public String root() {
        return "index";
    }
```

스프링 부트에서 정적 리소스 사용하기

- index.jsp

```
1  <%@ page language="java" contentType="text/html; charset=UTF-8"
2      pageEncoding="UTF-8"%>
3  <!DOCTYPE html>
4  <html>
5  <head>
6  <meta charset="UTF-8">
7  <title>Insert title here</title>
8  </head>
9  <body>
10     <h1>자기 소개</h1>
11     <button onclick="location.href='/name'">시작하기</button>
12 </body>
13 </html>
14
```

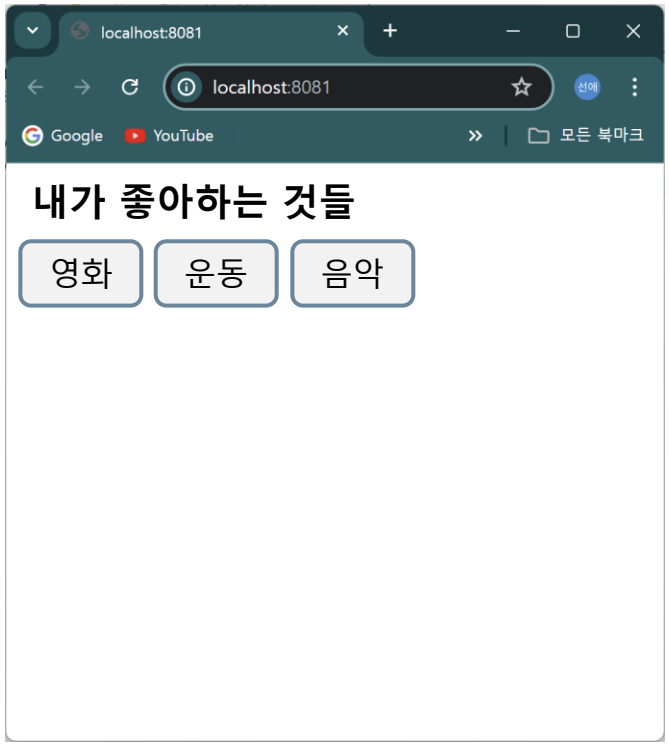
스프링 부트에서 정적 리소스 사용하기

- **좋아하는 것들 소개하기**
  - File > New > Spring Starter Project > **week10_03Favorite**
  - **프로젝트 생성 후 JSP 환경 설정**
  - **build.gradle**에 **디펜던시 추가**
  - **Gradle > Refresh Gradle Project**
  - 프로젝트 > properties > Project Facets
    - **Dynamic Web Module > 5.0**
  - **resources** 폴더
    - **application.properties** 수정
      - **spring.mvc.view.prefix=/WEB-INF/views/**
      - **spring.mvc.view.suffix=.jsp**
    - **JSP 폴더 생성**
      - src > main > **webapp > WEB-INF > views**
    - jsp 파일 생성
      - **views 폴더에 생성**

스프링 부트에서 정적 리소스 사용하기

- 프로젝트 생성 > **week10_ 03Favorite**
- 요청 및 응답 관계

```
@RequestMapping("/")
```




```
@RequestMapping("/movie")
```

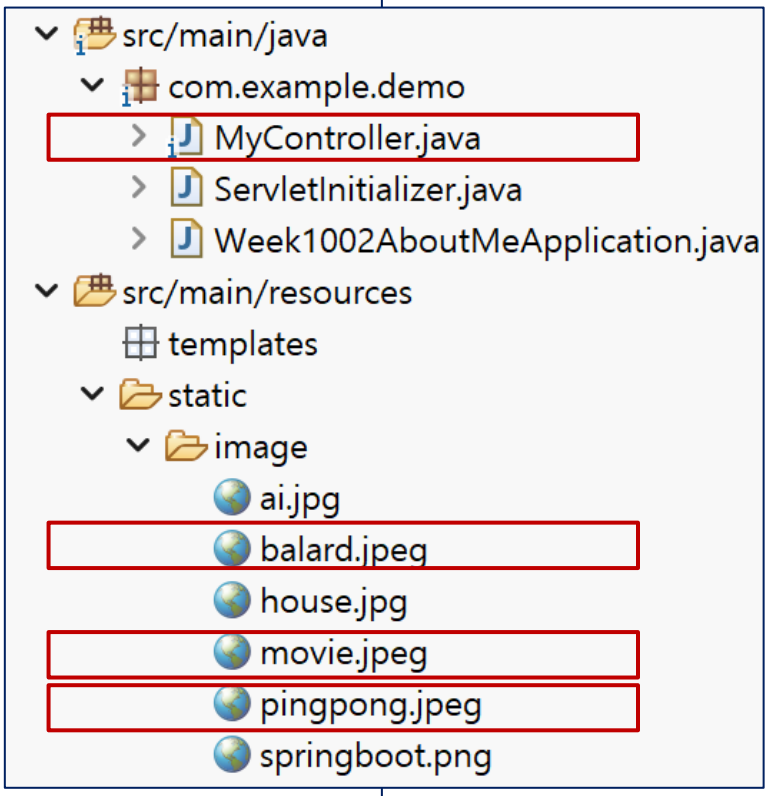
```
@RequestMapping("/sport")
```

```
@RequestMapping("/music")
```

스프링 부트에서 정적 리소스 사용하기

- 프로젝트 생성 > **week10_ 03Favorite**

```
@Controller
public class MyController {

    @RequestMapping("/")
    public String root() {
        return "index";
    }

    @RequestMapping("/movie")
    public String movie() {
        return "movie";
    }

    @RequestMapping("/sport")
    public String sport() {
        return "sport";
    }

    @RequestMapping("/music")
    public String music() {
        return "music";
    }
```



스프링 부트에서 정적 리소스 사용하기

- 프로젝트 생성 > **week10_ 03Favorite**

```
<body>
    <h1>내가 좋아하는 것들</h1>
    <button onclick="location.href='/movie'">영화</button>
    <button onclick="location.href='/sport'">운동</button>
    <button onclick="location.href='/music'">음악</button>
</body>
```

```
<body>
    <h1>영 화 : 터미네이터</h1>
    <img alt="" src="/image/movie.jpeg">
</body>
```

```
<body>
    <h1>운 동 : 탁구</h1>
    <img alt="" src="/image/pingpong.jpeg">
</body>
```

```
<body>
    <h1>음 악 : 발라드</h1>
    <img alt="" src="/image/balard.jpeg">
</body>
```

과제물

- **포켓몬 소개하기**
  - File > New > Spring Starter Project > **week10_04Pokemon**
  - **프로젝트 생성 후 JSP 환경 설정**
  - **build.gradle**에 **디펜던시 추가**
  - **Gradle > Refresh Gradle Project**
  - 프로젝트 > properties > Project Facets
    - **Dynamic Web Module > 5.0**
  - **resources** 폴더
    - **application.properties** 수정
      - **spring.mvc.view.prefix=/WEB-INF/views/**
      - **spring.mvc.view.suffix=.jsp**
    - **JSP 폴더 생성**
      - src > main > **webapp > WEB-INF > views**
    - jsp 파일 생성
      - **views 폴더에 생성**

과제물

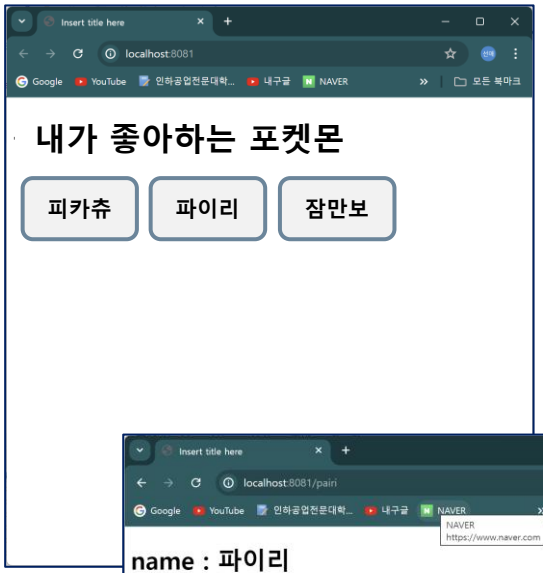
MyController 소스 코드와
최종 결과 화면(URL 포함)
이미지 파일제출

- **포켓몬 소개하기**
  - **MyController 클래스** 생성
  - URL 요청
    - "**/**" 요청
    - "**/pikachu**" 요청
    - "**/pairi**" 요청
    - "**/jammanbo**" 요청
  - **Model 객체** 이용해서 **name 정보** 전달하기

과제물

MyController 소스 코드와
최종 결과 화면(URL 포함)
이미지 파일제출

□ 포켓몬 소개하기



과제물

- **포켓몬 소개하기**
  - **MyController 클래스** 생성
  - URL 요청
    - "**/**" 요청
    - "**/pikachu**" 요청
    - "**/pairi**" 요청
    - "**/jammanbo**" 요청
  - **Model 객체** 이용해서 **name 정보** 전달하기

MyController 소스 코드와 최종 결과 화면(URL 포함) 이미지 파일제출

피카츄, 파이리, 잠만보를 제외한 3가지를 사용