1장. 스프링부트(SpringBoot) 개발 환경

스프링부트(SpringBoot)

스프링 부트(Spring Boot)

◆ 스프링부트(Spring Boot)란?

스프링 프레임워크의 서브 프로젝트로 만들어 졌으며, 이름에서 알 수 있듯 스프링과 부트의 합성어다.

부트(Boot)는 "컴퓨터를 부팅한다"는 말처럼 시스템을 사용 가능한 상태로 만드는 것을 의미하며, 웹 애플리케이션을 빠르게 제작하여 출시할 수 있는 경량의 오픈소스 프레임워크이다. 2014년 쯤 출시되었으며 점차 스프링에 비해 사용이 늘어나고 있는 추세이다.

- 스프링 부트의 특징

① **라이브러리 관리 자동화**

기존 자바프로젝트에서는 메이븐(Maven)이나 그레이들(Gradle)을 이용해서 라이브러리 의존성을 관리해왔으나, 스프링 부트에서는 스타터(Starter)라는 것을 이용해 특정 기능에 필요한 라이브러리 의존성을 더욱 간단히 처리할 수 있음

스프링 부트(Spring Boot)

② **설정의 자동화**

스프링부트에서는 프로젝트에 추가된 라이브러리를 기반으로 실행에 필요한 환경을 자동으로 설정해 준다. 라이브러리를 인지해 관련된 스프링 설정을 자동으로 처리해준다.

③ **테스트 환경과 내장 서버(톰캣)**

스프링부트로 생성한 프로젝트에는 Junit을 비롯한 테스트 관련 라이브러리들이 기본적으로 포함되어 있다. 또한 톰캣(Tomcat) 서버를 내장하고 있어서 main() 메소드를 가진 클래스를 실행하는 방식으로 구동하기 때문에 빠르게 실행 결과를 볼 수 있다.

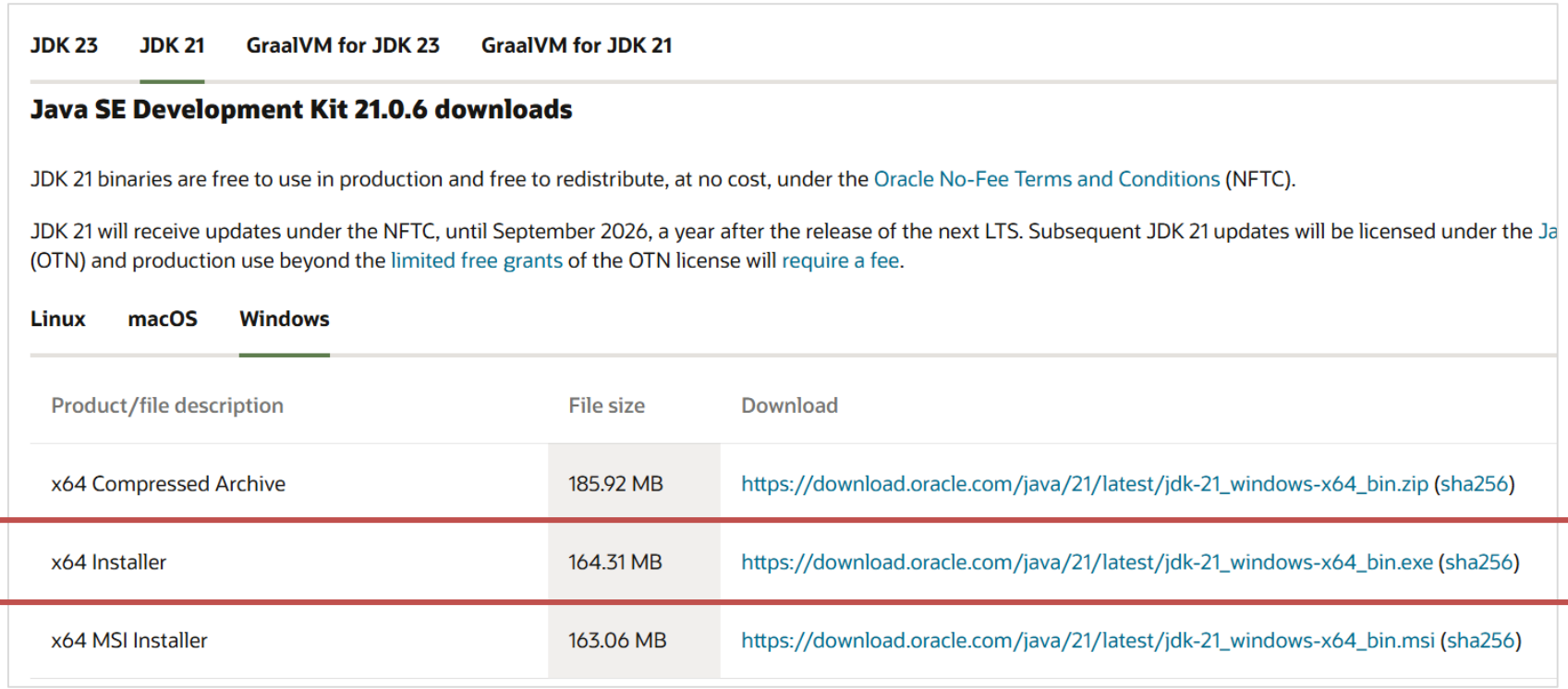
④ **독립적으로 실행 가능한 JAR**

애플리케이션을 실제 운영 서버에 배포하기 위해서 패키징(Packaging)을 해야하는데, WAR 파일이 아닌 JAR 파일로 패키징할 수 있다.

개발 환경 구축

◆ JDK 설치

- 오라클 java 다운로드(검색)-> windows> Java SE21 다운로드 -> x64 인스톨러



JDK 23 JDK 21 GraalVM for JDK 23 GraalVM for JDK 21

Java SE Development Kit 21.0.6 downloads

JDK 21 binaries are free to use in production and free to redistribute, at no cost, under the Oracle No-Fee Terms and Conditions (NFTC).

JDK 21 will receive updates under the NFTC, until September 2026, a year after the release of the next LTS. Subsequent JDK 21 updates will be licensed under the Ja
(OTN) and production use beyond the limited free grants of the OTN license will require a fee.

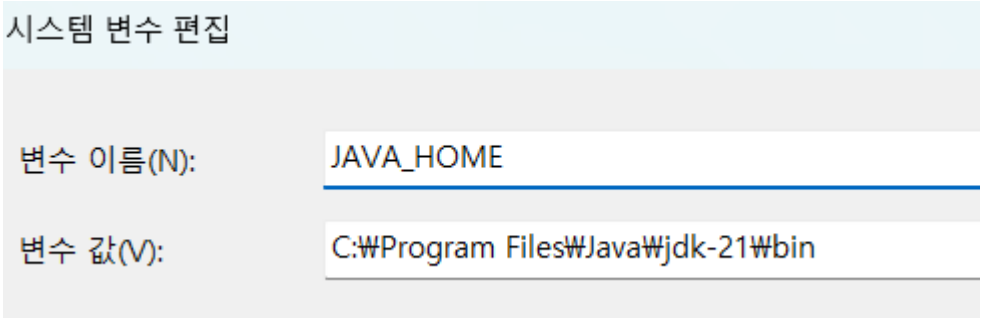
Linux macOS Windows

Product/file description	File size	Download
x64 Compressed Archive	185.92 MB	https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.zip (sha256)
x64 Installer	164.31 MB	https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.exe (sha256)
x64 MSI Installer	163.06 MB	https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.msi (sha256)

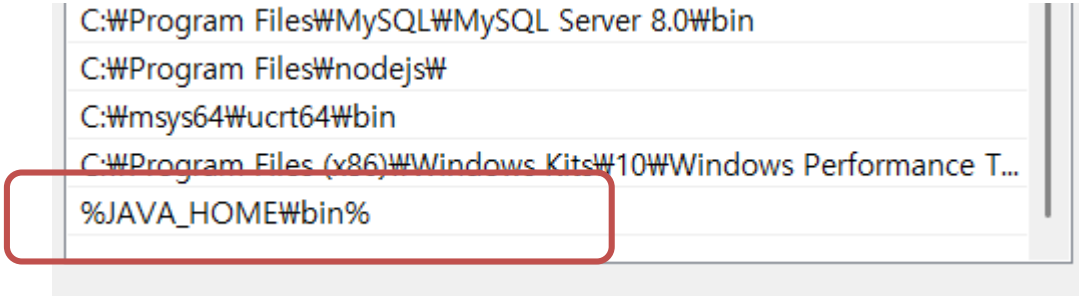
개발 환경 구축

◆ 환경 변수 설정

1. 시스템 변수> 새로 만들기 > JAVA_HOME 만들기

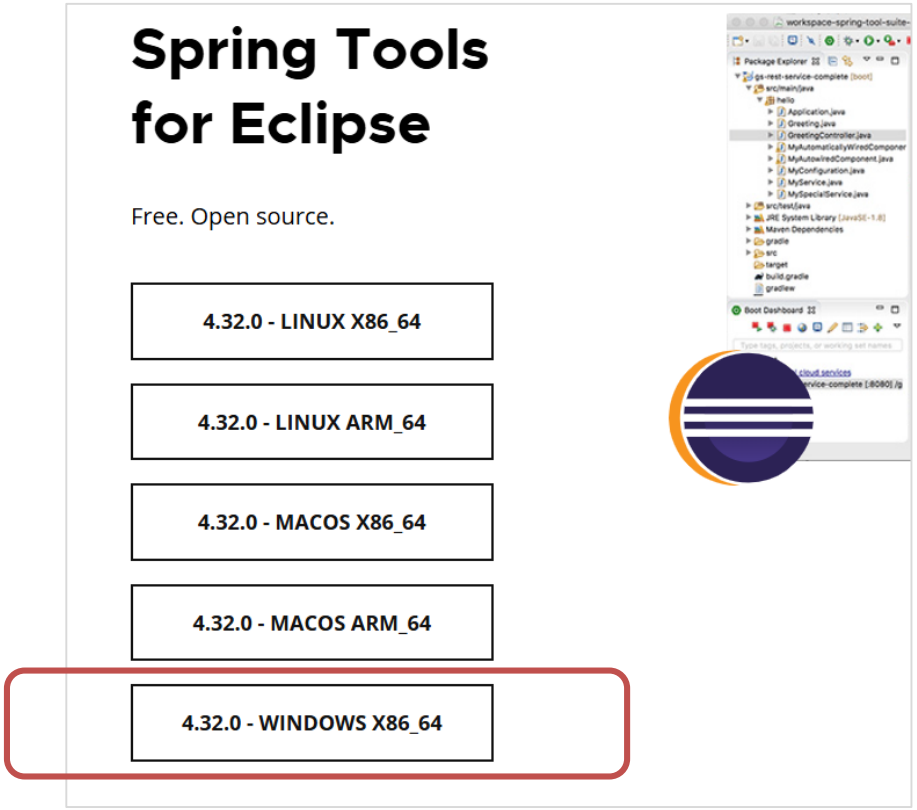


2. 시스템 변수> path > %JAVA_HOME₩bin% 추가



STS4(Spring Tool Suite)

◆ STS4(Spring Tool Suite) 설치

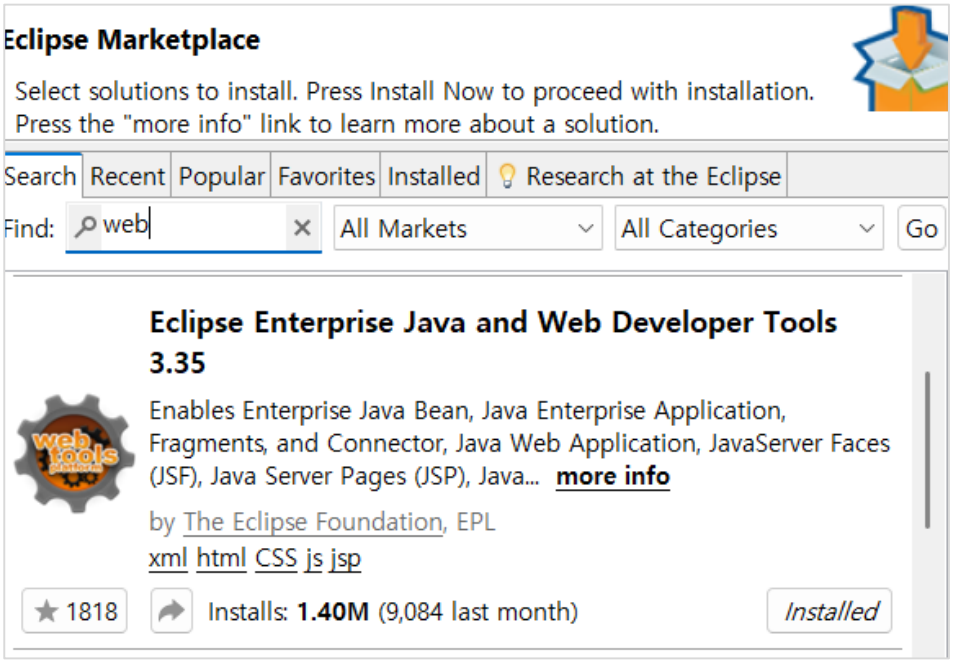


STS4(Spring Tool Suite)

◆ Web Developer Tools 설치

Help > Eclipse Marketplace > web 검색 > Java and Web 설치

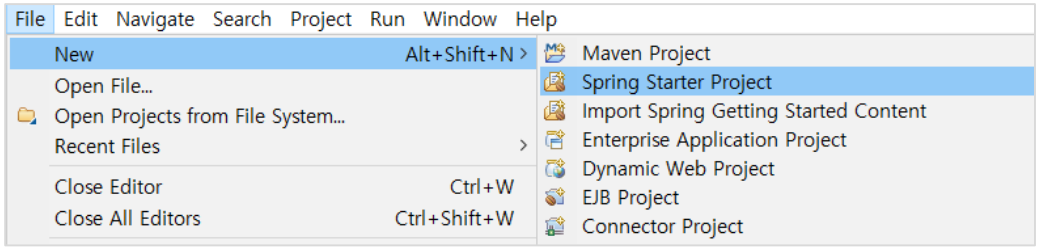
설치하지 않으면 html, css, js 파일 등이 지원되지 않음



jwboot 프로젝트

- 실습 프로젝트 생성

  [File] -> [New] -> [Spring Starter Project]



Name-jwboot

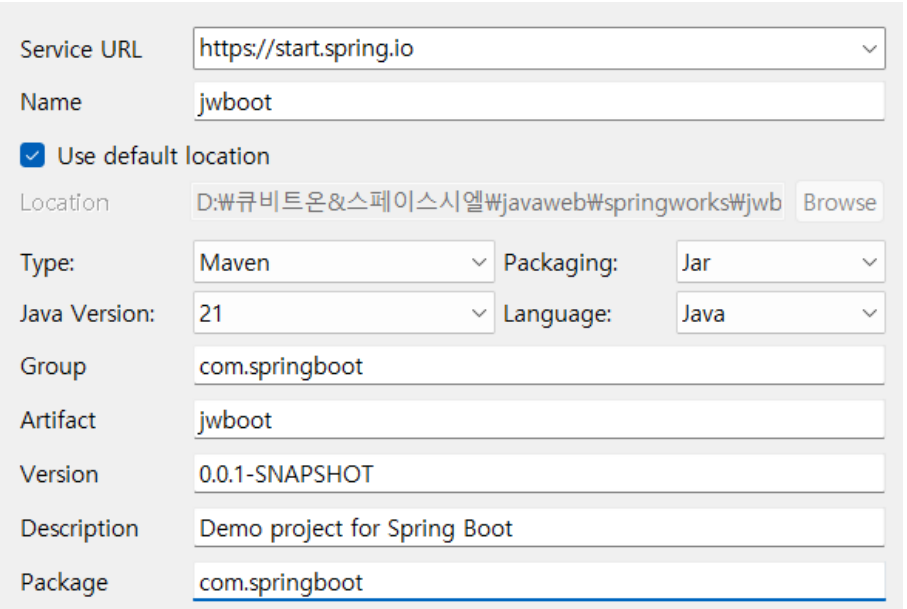
Type-Maven

Packing-Jar

Java Version-21

Group-com.space

Package-com.space

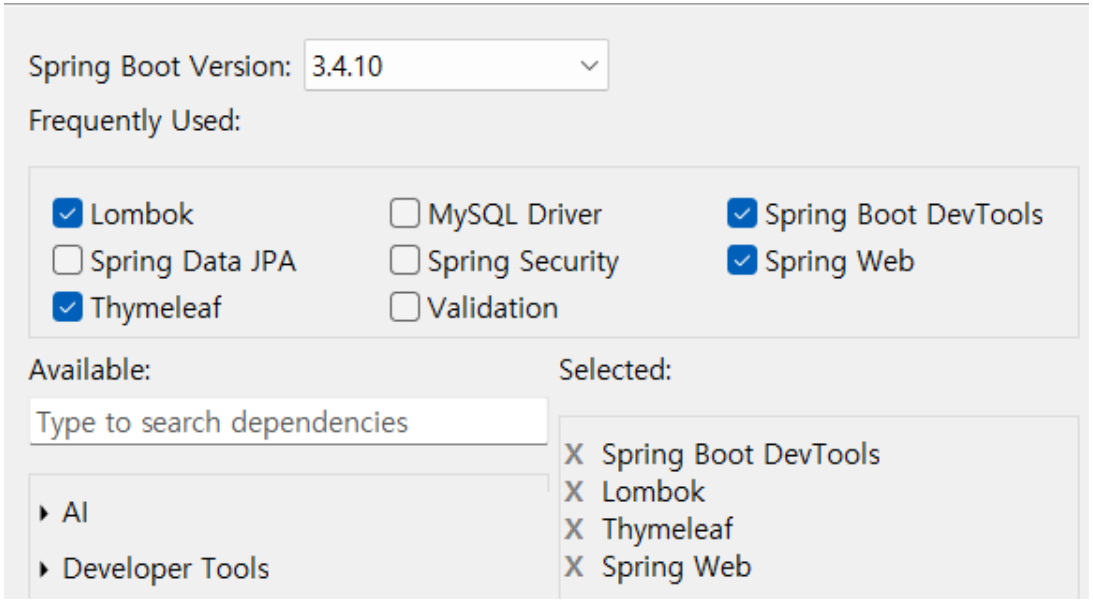


jwboot 프로젝트

- 실습 프로젝트 생성

Spring Boot Version – 3.4.10

Lombok, Spring Web, Spring Boot DevTools, Thymeleaf 모듈 추가



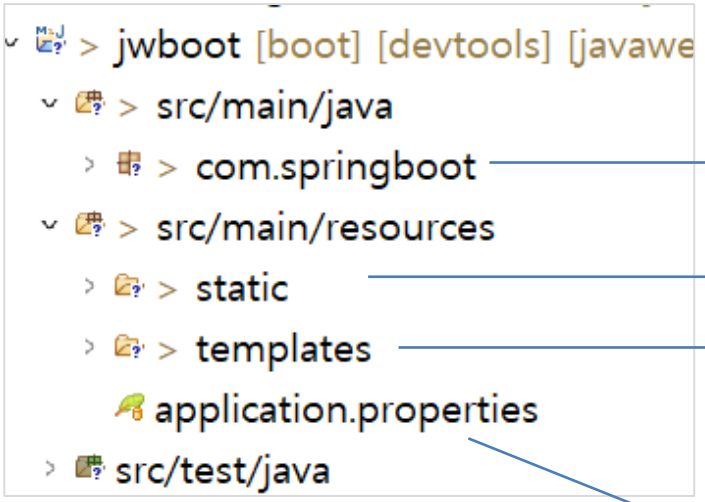
jwboot 프로젝트

- pom.xml

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <scope>runtime</scope>
        <optional>true</optional>
    </dependency>
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <optional>true</optional>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>
```

jwboot 프로젝트

- **프로젝트 구조(프레임 워크 구조)**



Java 파일

정적 파일 : css, js, image

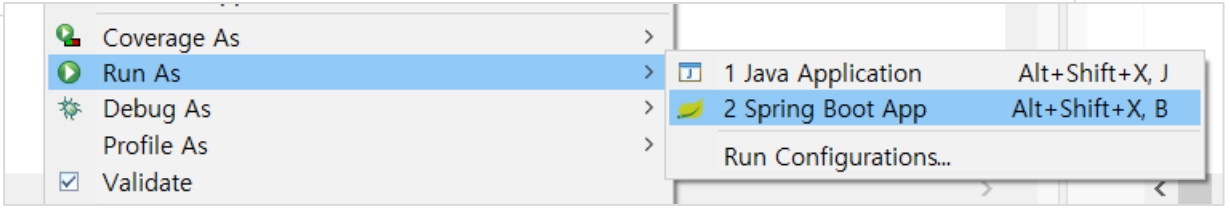
동적 : html 파일

설정 파일: 서버설정, DB 연결 등

jwboot 프로젝트

- SpringBoot App으로 실행

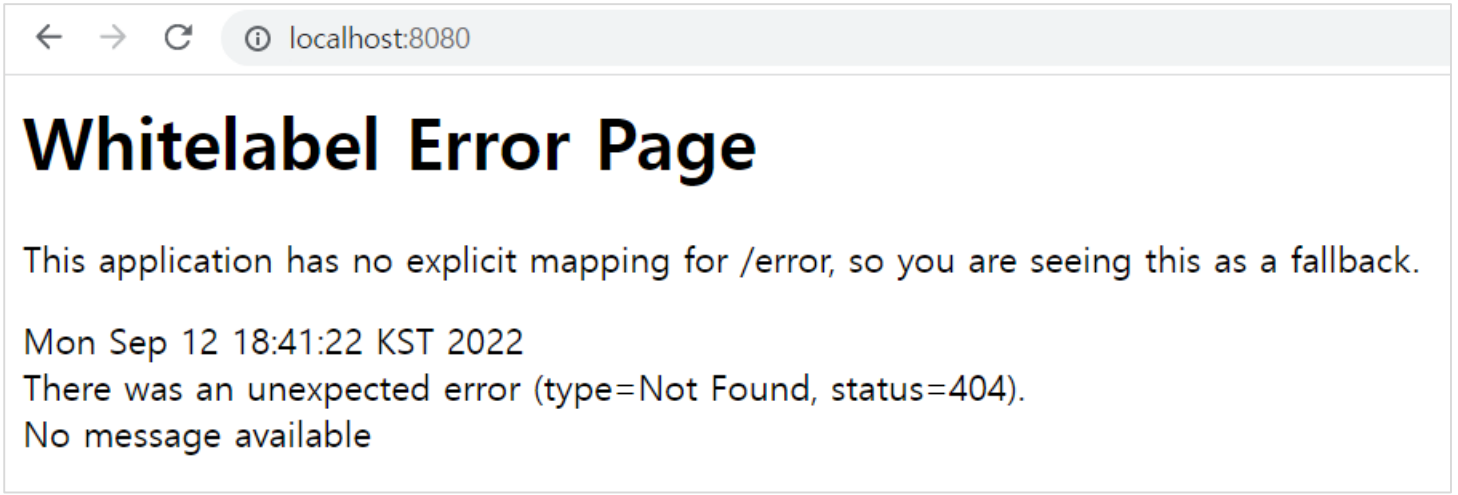
```java
@SpringBootApplication
public class Chapter01Application {

    public static void main(String[] args) {
        //Spring Boot App으로 실행
        SpringApplication.run(Chapter01Application.class, args);
    }
}
```




```
2024-01-11T06:01:45.691+09:00  INFO 4900 --- [  restartedMain] com.kh.study.JwbootApplication
2024-01-11T06:02:01.549+09:00  INFO 4900 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]
2024-01-11T06:02:01.550+09:00  INFO 4900 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet
2024-01-11T06:02:01.551+09:00  INFO 4900 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet
```

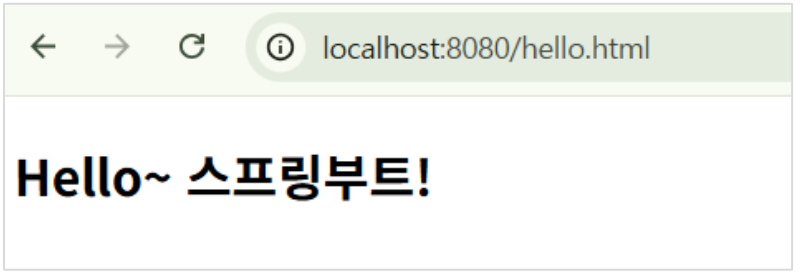
jwboot 프로젝트

- http://localhost:8080



jwboot 프로젝트

- **static – 정적 파일**



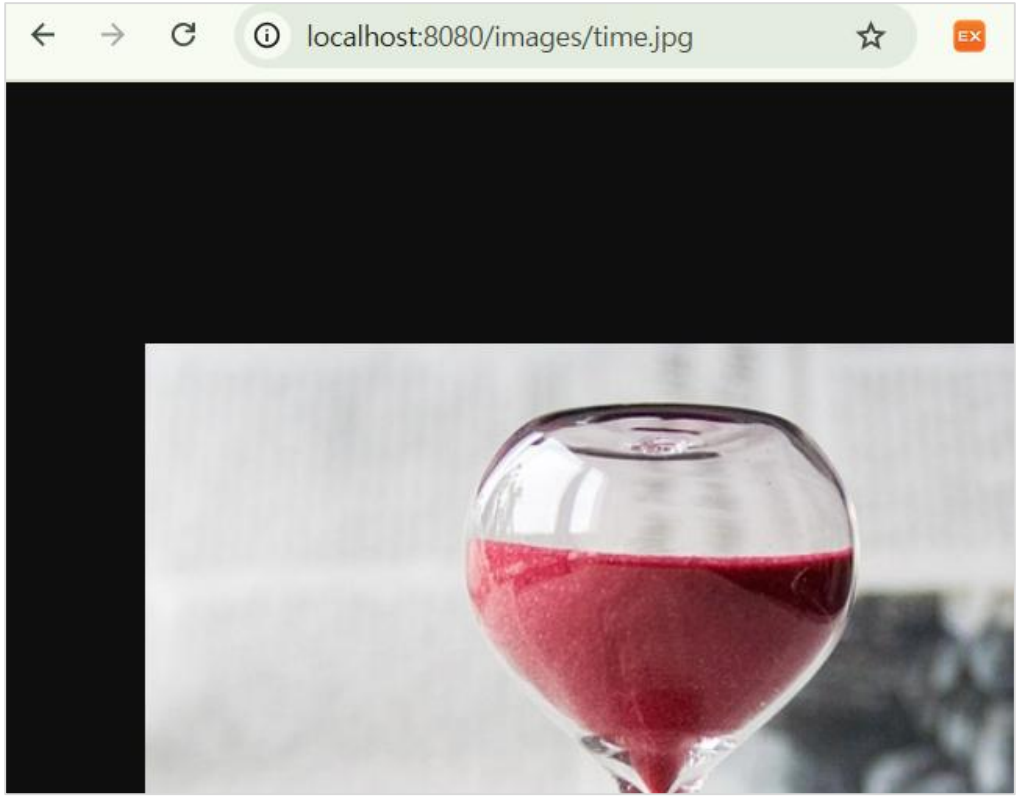
static > hello.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>hello~</title>
</head>
<body>
    <h2>Hello~ 스프링부트!</h2>
</body>
</html>
```

jwboot 프로젝트

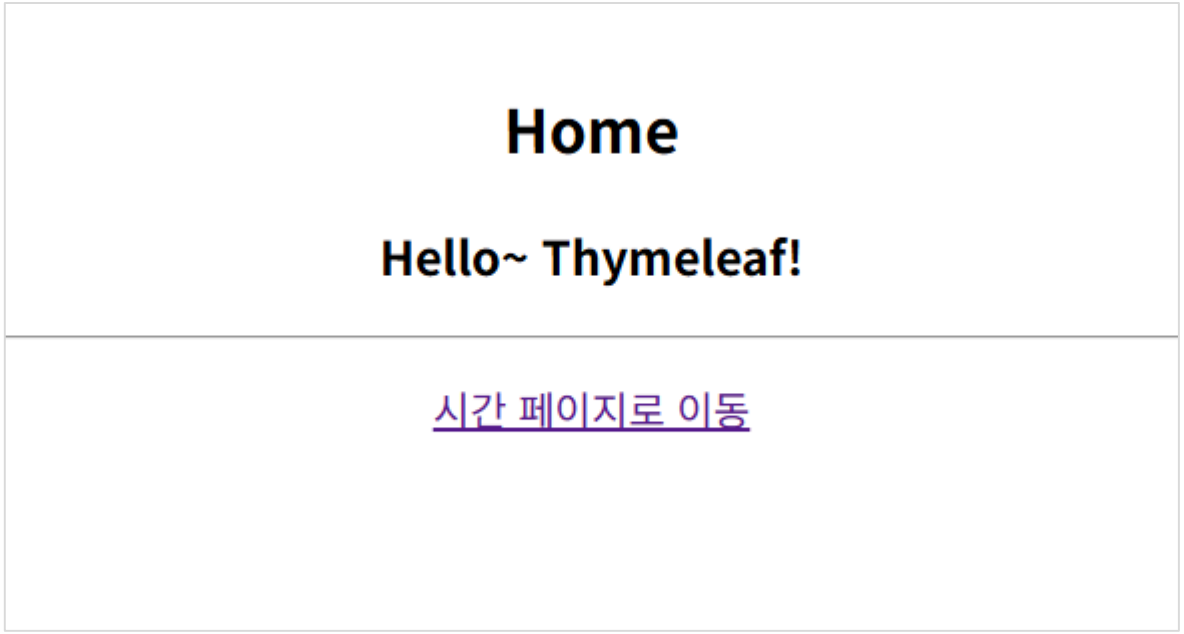
- **static – 정적 파일**

  static > images > time.jpg



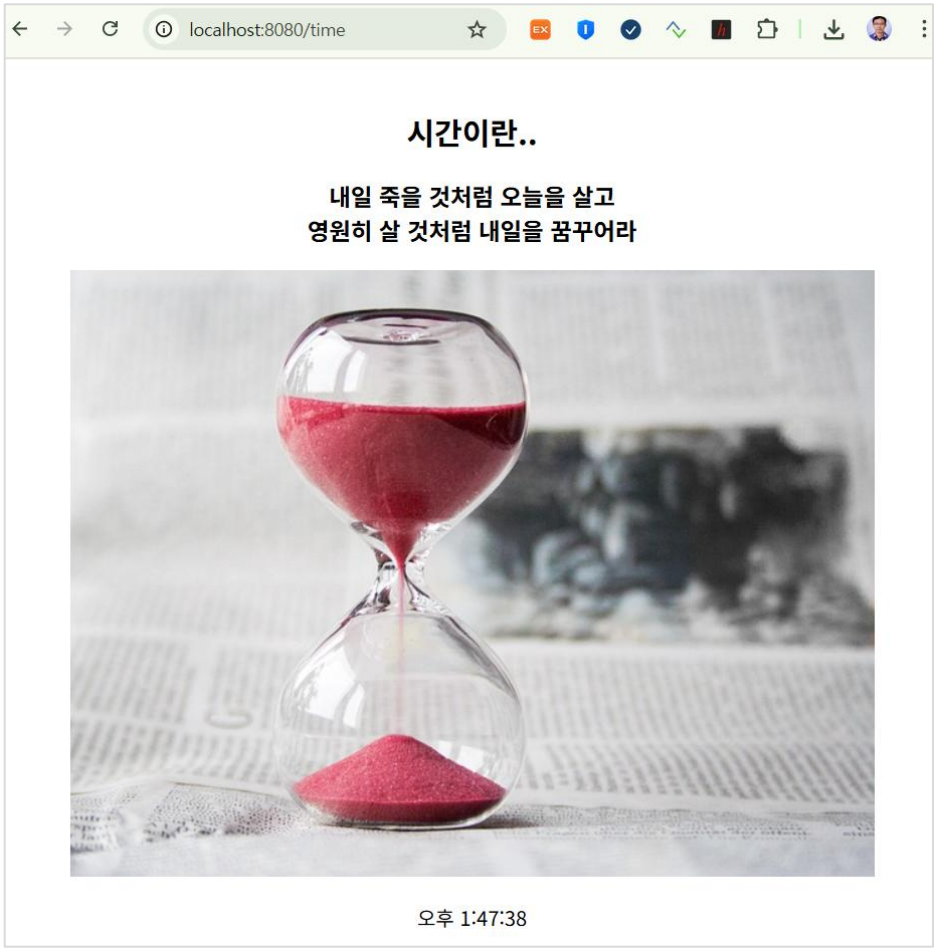
메인 페이지 만들기

- 메인 페이지 만들기



메인 페이지 만들기

- 메인 페이지 만들기



메인 페이지 만들기

- **url 매핑 - HomeController**

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

@Controller
public class HomeController {

    @GetMapping("/")
    public String home() {
        return "home";
    }

    @GetMapping("/time")
    public String time() {
        return "/pages/time";
    }
}
```

메인 페이지 만들기

- **templates > home.html**

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>home</title>
<link rel="stylesheet" href="/css/style.css">
</head>
<body>
    <div id="content">
        <h2>홈페이지 방문을 진심으로 환영합니다.</h2>
        <hr>

        <a href="/time">시간 페이지로 이동</a>
    </div>
</body>
</html>
```

메인 페이지 만들기

- **templates > pages > time.html**

```
<head>
<meta charset="UTF-8">
<title>시간 이란..</title>
<link rel="stylesheet" href="/css/style.css">
<script src="/js/time.js"></script>
</head>
<body>
    <div id="content">
        <h2>시간이란..</h2>
        <h3>내일 죽을 것처럼 오늘을 살고<br>
            영원히 살 것처럼 내일을 꿈꾸어라
        </h3>
        <img alt="시계" src="/images/time.jpg">
        <p id="display">
    </div>
</body>
</html>
```

메인 페이지 만들기

- static > css > style.css

```css
@charset "UTF-8";

#content{
	width: 90%;
	margin: 40px auto;
	text-align: center;
}
```

메인 페이지 만들기

- **static > js > time.js**

```
/*setInterval(myWatch, 1000);

function myWatch(){
    let date = new Date();
    let now = date.toLocaleTimeString();
    document.getElementById("display").innerHTML = now;
}*/

setInterval(()=>{
    let date = new Date();
    let now = date.toLocaleTimeString();
    document.getElementById("display").innerHTML = now;
}, 1000);
```

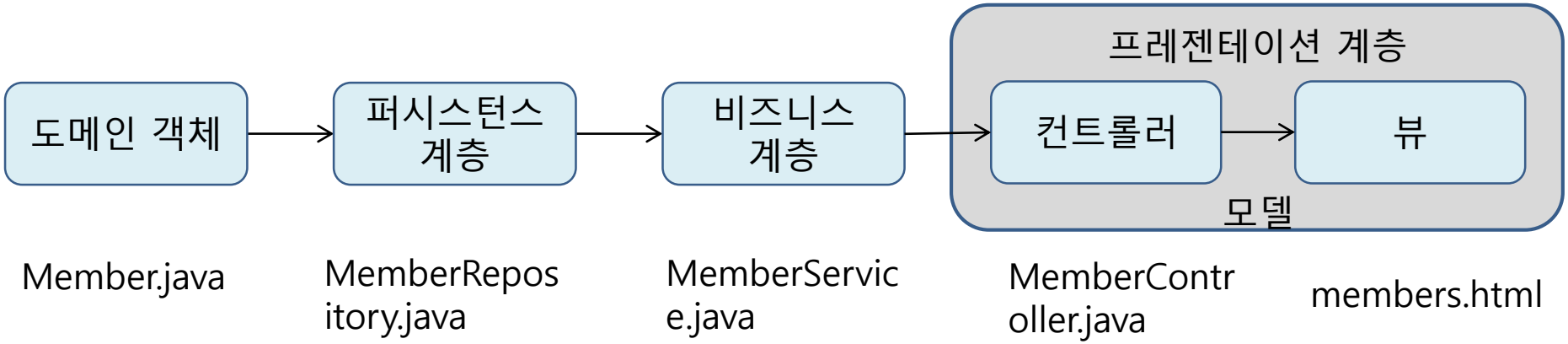
메인 페이지 만들기

- 컨트롤러 사용한 URL 매핑

```java
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class HomeController {

    @GetMapping("/")
    public String home(Model model) {
        //모델(message)에 문자열을 담아 보내기
        model.addAttribute("message", "Hello~ Thymeleaf!");
        return "home"; //파일: home.html
    }

    @GetMapping("/time")
    public String time() {
        return "/pages/time";  //파일경로: /pages/home.html
    }
}
```

스프링 MVC

- **계층적 구조의 구현 과정**



스프링 MVC

- **스프링 MVC의 3 계층 구조**

계층	어노테이션	역할
Controller	@Controller	클라이언트 요청처리, 화면 반환
Service	@Service	비즈니스 로직 수행
Repository	@Repository	데이터베이스 접근 CRUD 처리

스프링 MVC

- **컨트롤러란?**

  스프링 MVC 패턴과 연동하는 스프링 부트가 웹에서 사용자 요청을 받으면 가장 먼저 내장된 톰캣 서버의 디스패처 서블릿으로 전달한다.
  디스패처 서블릿은 요청을 받는 창구 역할만 할뿐 실제 처리는 컨트롤러(controller)가 한다.

- **@Controller 사용**

  @Controller는 스프링 MVC 프레임워크에서 사용자의 요청을 받아 웹 브라우저에 보여줄 HTML을 뷰(View)로 보내는 역할을 하는 클래스를 지정하는 어노테이션이다.
  클라이언트로부터 HTTP 요청을 받으면, 이를 처리하기 위해 서비스 계층으로 위임하고, 처리 결과를 '뷰 리졸버(View Resolver)'를 통해 HTML 파일을 찾아 응답으로 반환한다.

스프링 MVC

- @Repository 사용

@Repository는 Spring에서 데이터 접근 계층(DAO, Repository)을 나타내는 어노테이션(Annotation)이다.
즉, DB와 직접 통신하는 클래스임을 스프링에게 알려주는 표시입니다.

- 빈(Bean) 등록

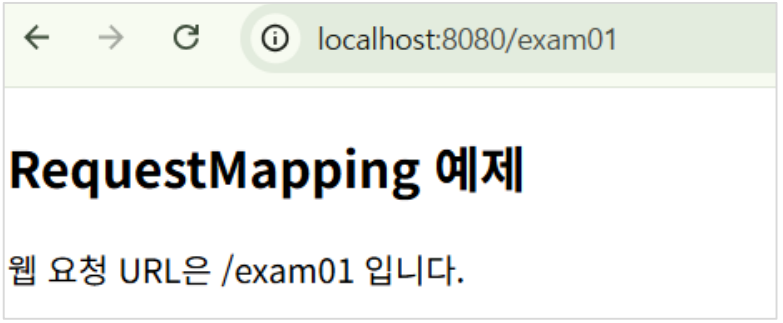
@Repository가 붙은 클래스는 Spring이 자동으로 감지해서 스프링 컨테이너에 빈으로 등록한다.
따라서 @Autowired나 생성자 주입을 통해 서비스나 컨트롤러에서 바로 사용할 수 있다.

스프링 MVC

- **@RequestMapping을 이용한 요청 매칭 경로 설정**

@RequestMapping(value="URL", method=RequestMethod.HTTP 요청)

@RequestMapping("URL") (단축형)



스프링 MVC

- **@RequestMapping을 이용한 요청 매칭 경로 설정**

```java
@Controller
public class Exam01Controller {

    //@RequestMapping(value="/exam01", method=RequestMethod.GET)
    @RequestMapping("/exam01")
    public String requestMethod() {
        return "pages/view01";  //view01.html
    }
}
```

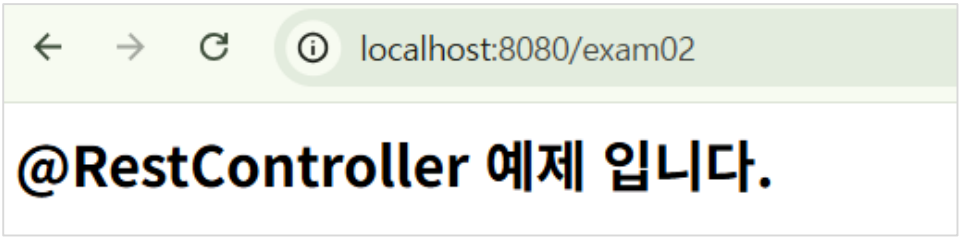
```
<title>RequestMapping 예제</title>
</head>
<body>
    <h2>RequestMapping 예제</h2>
    <p>웹 요청 URL은 /exam01 입니다.</p>
</body>
</html>
```

pages/view01.html

스프링 MVC

- **@RestController를 이용한 컨트롤러 정의**

  @RestController는 Spring MVC에서 RESTful 웹 서비스(API)를 만들 때 사용하는 핵심 어노테이션이다.
  웹 화면(HTML)을 반환하는 @Controller + Thymeleaf와는 달리,
  @RestController는 데이터(JSON/XML)를 직접 반환하는 컨트롤러이다.



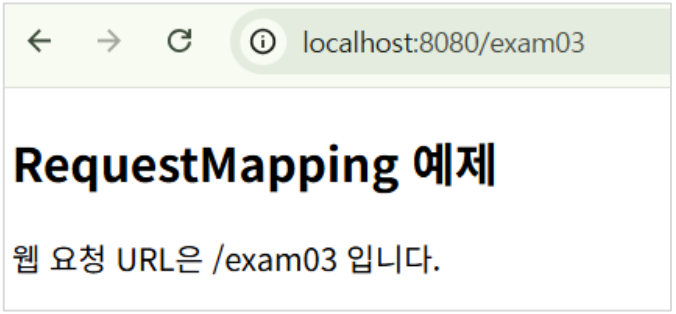
스프링 MVC

- **@RestController를 이용한 컨트롤러 정의**

```
@RequestMapping("/exam02")
@RestController
public class Exam02Controller {

    @RequestMapping
    public String requestMethod() {
        return "<h2>@RestController 예제 입니다.</h2>";
    }
}
```

스프링 MVC

- **@GetMapping를 이용한 컨트롤러 정의**




```java
@Controller
public class Exam03Controller {

    @GetMapping("/exam03")
    public String requestMethod() {
        return "pages/view02";  //view02.html
    }
}
```

pages/view02.html