

Thymeleaf



Soft Engineer Society

▣ 실습 환경

- 실습환경

- Spring tool suite (URL : <https://spring.io/>)
- Chrome

- 참고 사이트

- URL : <https://www.thymeleaf.org/>



▣ Thymeleaf 특징

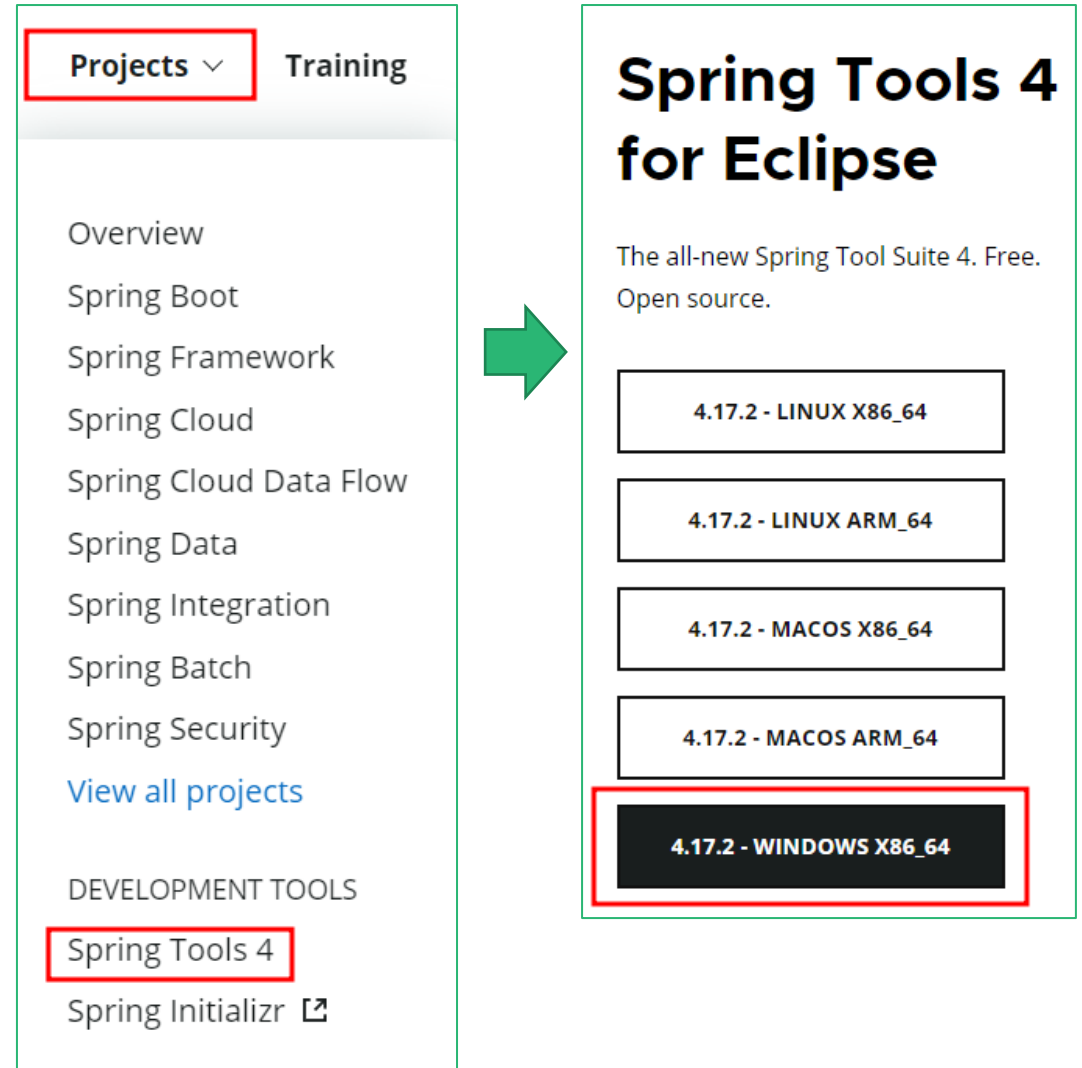
• 특징

- Thymeleaf는 HTML, XML, JavaScript, CSS 및 일반 텍스트까지 처리할 수 있는 웹 및 독립 실행형 환경을 위한 최신 서버 측 Java Template Engine이다.
- 확장명이 .html이므로 문서 자체로 브라우저에 표시 가능하고, 정적 프로토타입으로도 동작한다.
- JSP와 같은 별도의 태그가 존재하지 않는 Natural Templates의 개념을 기반으로 작성되었다.
- HTML5를 염두에 두고 설계되어 있으므로 필요한 경우 완전히 검증된 템플릿을 생성할 수 있다.
- 유지관리가 쉽고, 디자인 프로토타입으로 사용되는 디자인 템플릿에 영향을 미치지 않기 때문에 디자이너와 개발자가 협업하기 좋다.
- Spring Boot에서 기본적으로 Thymeleaf를 템플릿 엔진으로 사용하므로 dependency 추가 이외 별도의 설정이 필요 없다.
- 현재 버전 : 3.0

□ 환경 설정

• Spring Tool Suite 다운로드

- URL : <https://spring.io/>
- 위 사이트의 상단 메뉴 중
[Project]-[Spring Tools 4]-[Spring Tools 4 for Eclipse]
에서 자신의 컴퓨터 운영체제에 맞는 프로그램을 다운로드



Projects ▾ Training

Overview
Spring Boot
Spring Framework
Spring Cloud
Spring Cloud Data Flow
Spring Data
Spring Integration
Spring Batch
Spring Security
[View all projects](#)

DEVELOPMENT TOOLS
Spring Tools 4
Spring Initializr ↗

Spring Tools 4 for Eclipse

The all-new Spring Tool Suite 4. Free. Open source.

4.17.2 - LINUX X86_64

4.17.2 - LINUX ARM_64

4.17.2 - MACOS X86_64

4.17.2 - MACOS ARM_64

4.17.2 - WINDOWS X86_64

▣ 설치 및 실행

• 압축 해제

- ① 명령 프롬프트를 실행 후 위에서 다운로드 받은 파일이 있는 디렉토리로 이동
- ② 아래의 명령어를 입력하면 해당 디렉토리에 압축이 해제된다.

```
C:\W>java -jar spring-tool-suite-4-4.17.2.RELEASE-e4.26.0-win32.win32.x86_64.self-extracting.jar
```

• 실행 파일

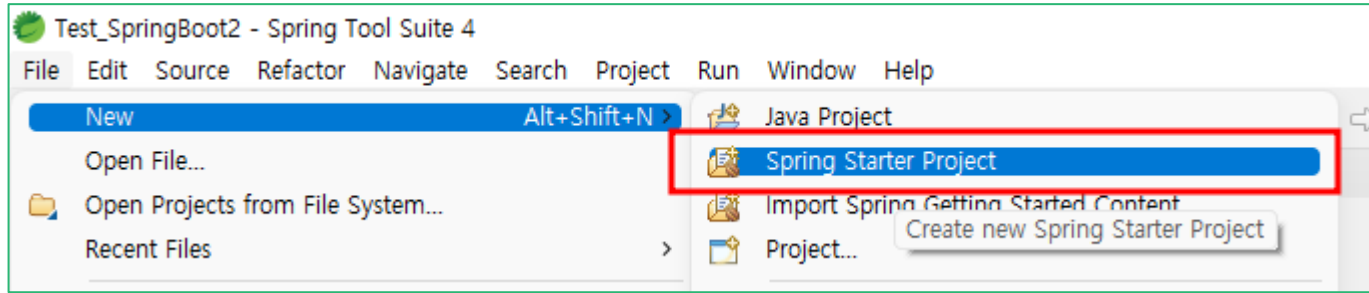
- 압축 해제된 디렉토리 내 “SpringToolSuite4.exe” 파일

이름	수정한 날짜	유형	크기
configuration	2022-03-02 오전 11:08	파일 폴더	
dropins	2022-03-02 오전 11:08	파일 폴더	
features	2022-03-02 오전 11:08	파일 폴더	
META-INF	2022-03-02 오전 11:08	파일 폴더	
p2	2022-03-02 오전 11:08	파일 폴더	
plugins	2022-03-02 오전 11:08	파일 폴더	
readme	2022-03-02 오전 11:08	파일 폴더	
.eclipseproduct	2022-03-02 오전 11:08	ECLIPSEPRODUCT...	1KB
artifacts.xml	2022-03-02 오전 11:08	Microsoft Edge H...	158KB
eclipse.exe	2022-03-02 오전 11:08	응용 프로그램	231KB
license.txt	2022-03-02 오전 11:08	텍스트 문서	12KB
open-source-licenses.txt	2022-03-02 오전 11:08	텍스트 문서	837KB
SpringToolSuite4.exe	2022-03-02 오전 11:08	응용 프로그램	518KB
SpringToolSuite4.ini	2022-03-02 오전 11:08	구성 설정	1KB

□ Spring boot project 생성

- 프로젝트 생성

- STS를 구동한 뒤 [File]-[New]-[Spring Starter Project] 실행



설정 (1단계) : 환경 구성

New Spring Starter Project

Service URL

https://start.spring.io

Name

web-1

☒ Use default location

Location

C:\workspace\Test_Spring\web-1

Browse

Type:

Gradle - Groovy

Packaging:

Jar

Java Version:

8

Language:

Java

Group

net.softsociety.net

Artifact

web-1

Version

0.0.1-SNAPSHOT

Description

Thymeleaf 문법의 이해

Package

net.softsoceity.web-1

Working sets

☐ Add project to working sets

New...

Working sets:

Select...

?

< Back

Next >

Finish

Cancel

설정	내용
Type	Gradle - Groovy
Packing	Jar
Java Version	8
Language	Java
Group	net.softsociety
Artifact	web
Package	net.softsociety.web

□ 설정 (2단계) : Dependency 추가

New Spring Starter Project

Service URL:

Name:

☒ Use default location

Location:

Type: Packaging:

Java Version: Language:

Group:

Artifact:

Version:

Description:

Package:

Working sets

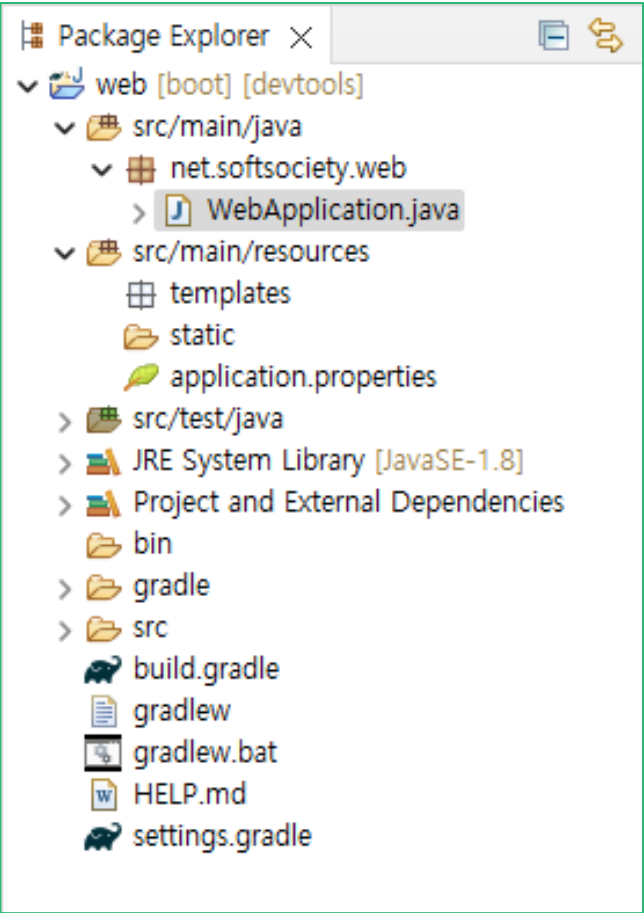
☐ Add project to working sets

Working sets:

설정	내용
Spring Boot Version	2.7.9 (2023년 3월 현재 stable 버전)
추가 Dependency	Thymeleaf
	Spring Web
	Lombok
	Spring Boot DevTools

Project 구조

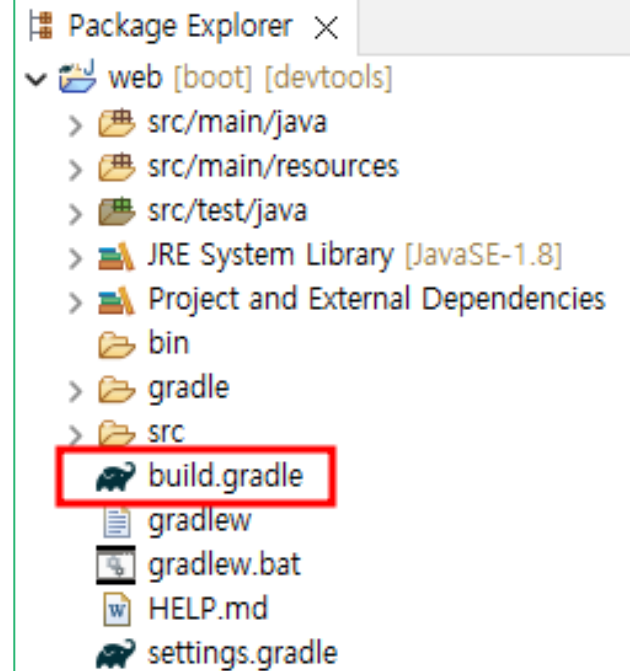
- Project Structure 구성이 완료되면 아래와 같은 구조의 프로젝트가 생성된다.



주요 디렉토리와 파일		설명
src/ <u>main</u> /java	net.softsociety.web	Server-side 코드 작성 위치
	WebApplication.java	프로그램 구동에 필요한 main() 메소드
src/main/ <u>resources</u>	templates	Client-side html 문서 위치
	static	정적 파일(image, css, javascript)
	application.properties	프로젝트 운영에 필요한 설정 정보 (Port, DB 설정 등)
src/ <u>test</u> /java	단위 테스트에 필요한 코드 작성 위치	
build.gradle	프로젝트 생성 시 등록된 설정 정보	

■ build.gradle (1)

```
plugins {  
    id 'org.springframework.boot' version '2.7.9'  
    id 'io.spring.dependency-management' version '1.0.15.RELEASE'  
    id 'java'  
}  
  
group = 'net.softsociety'  
version = '0.0.1-SNAPSHOT'  
sourceCompatibility = '1.8'  
  
configurations {  
    compileOnly {  
        extendsFrom annotationProcessor  
    }  
}  
  
repositories {  
    mavenCentral()  
}
```



▣ build.gradle (2)

```
dependencies {  
    implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'  
    implementation 'org.springframework.boot:spring-boot-starter-web'  
    compileOnly 'org.projectlombok:lombok'  
    developmentOnly 'org.springframework.boot:spring-boot-devtools'  
    annotationProcessor 'org.projectlombok:lombok'  
    testImplementation 'org.springframework.boot:spring-boot-starter-test'  
}  
  
tasks.named('test') {  
    useJUnitPlatform()  
}
```

▣ application.properties (1)

- 주요 설정 정보

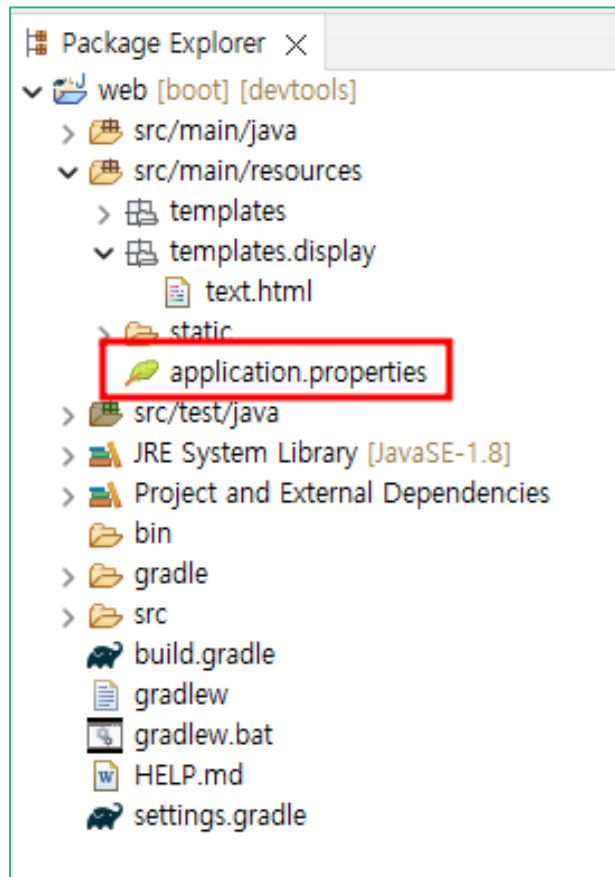
- 필요한 정보는 필요할 때마다 추가하여 사용할 수 있다.

서버 포트 번호 변경

server.port=8080

context path 설정 : default context path /

server.servlet.context-path=/sampleBoard



▣ application.properties (2)

• 주요 설정 정보

오라클 연결 설정

```
spring.datasource.driver-class-name=oracle.jdbc.OracleDriver  
spring.datasource.url=jdbc:oracle:thin:@localhost:1521/x  
spring.datasource.username=hr  
spring.datasource.password=hr
```

MyBatis type-alias 설정

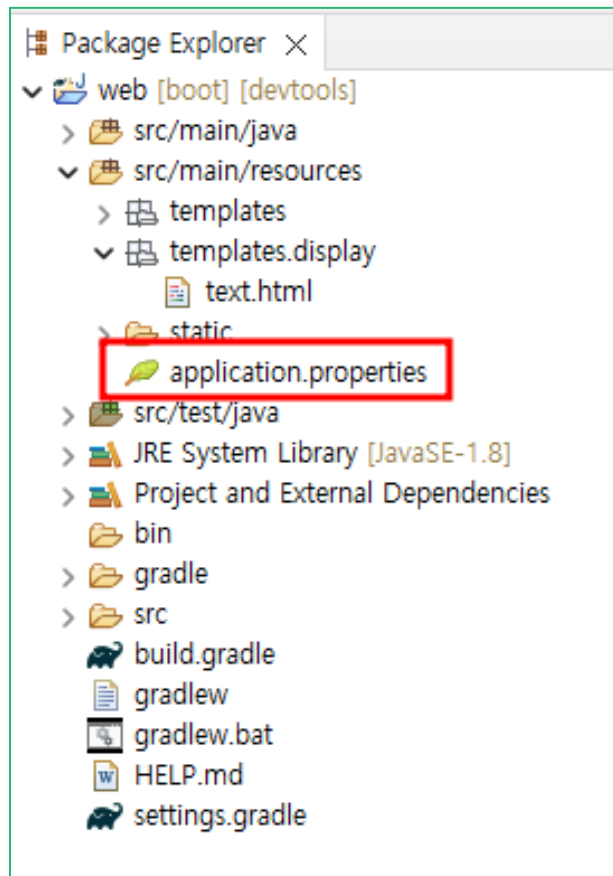
```
mybatis.type-aliases-package=org.smart.board.entity
```

Mybatis mapper 위치 설정

```
mybatis.mapper-locations=mappers/**/*.xml
```

업로드 용량 제한: default 1M(1048576)

```
spring.servlet.multipart.maxFileSize=100MB  
spring.servlet.multipart.maxRequestSize=100MB  
spring.servlet.multipart.location=c:/upload
```



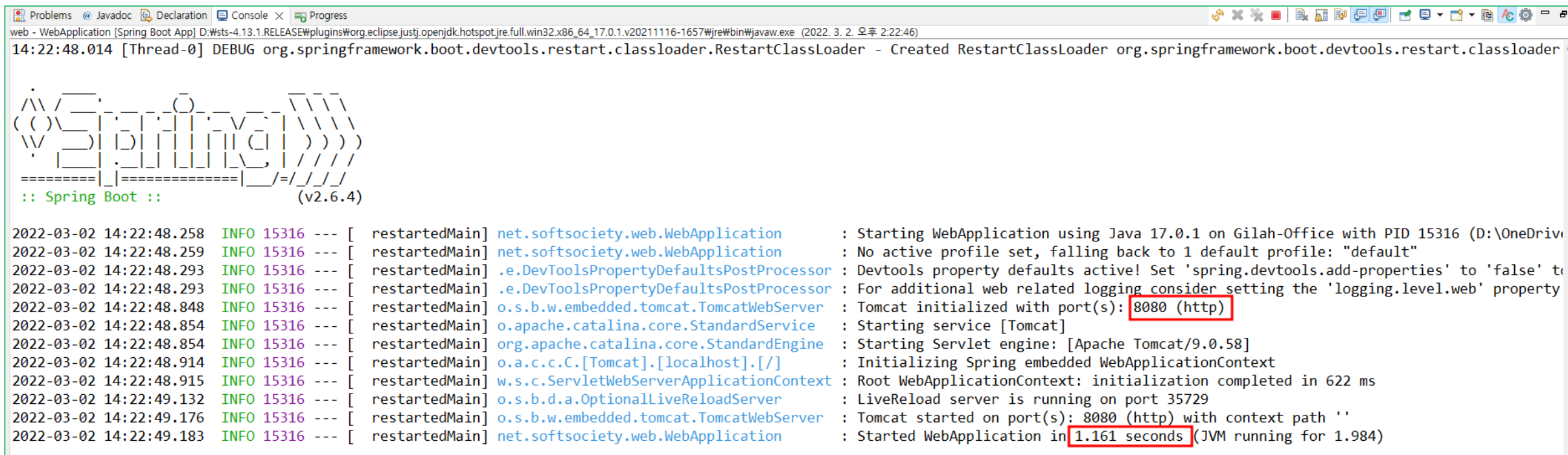
□ Run (1)

• 프로그램의 구동

- main() 메소드가 포함되어 있는 xxxApplication.java 파일을 오픈한 후 **[Run]-[Run As]-[Spring Boot App]** 실행

• 구동 결과

- 아래의 화면은 내장된 Tomcat에 의해 서버가 올바르게 구동되었을 때의 모습이다



```
web - WebApplication [Spring Boot App] D:\wsts-4.13.1.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.1.v20211116-1657\jre\bin\javaw.exe (2022. 3. 2. 오후 2:22:46)
14:22:48.014 [Thread-0] DEBUG org.springframework.boot.devtools.restart.classloader.RestartClassLoader - Created RestartClassLoader org.springframework.boot.devtools.restart.classloader

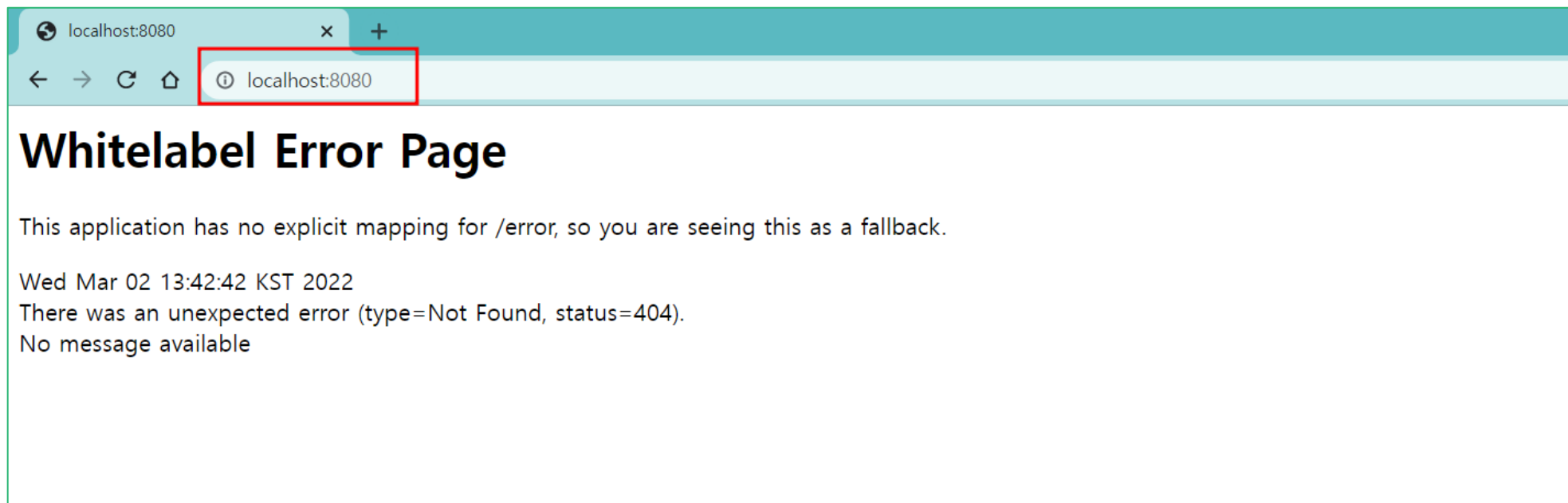
:: Spring Boot :: (v2.6.4)

2022-03-02 14:22:48.258 INFO 15316 --- [ restartedMain] net.softsociety.web.WebApplication : Starting WebApplication using Java 17.0.1 on Gilah-Office with PID 15316 (D:\OneDrive
2022-03-02 14:22:48.259 INFO 15316 --- [ restartedMain] net.softsociety.web.WebApplication : No active profile set, falling back to 1 default profile: "default"
2022-03-02 14:22:48.293 INFO 15316 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.devtools.add-properties' to 'false' to
2022-03-02 14:22:48.293 INFO 15316 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting the 'logging.level.web' property
2022-03-02 14:22:48.848 INFO 15316 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2022-03-02 14:22:48.854 INFO 15316 --- [ restartedMain] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2022-03-02 14:22:48.854 INFO 15316 --- [ restartedMain] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.58]
2022-03-02 14:22:48.914 INFO 15316 --- [ restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2022-03-02 14:22:48.915 INFO 15316 --- [ restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 622 ms
2022-03-02 14:22:49.132 INFO 15316 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
2022-03-02 14:22:49.176 INFO 15316 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2022-03-02 14:22:49.183 INFO 15316 --- [ restartedMain] net.softsociety.web.WebApplication : Started WebApplication in 1.161 seconds (JVM running for 1.984)
```

□ Run (2)

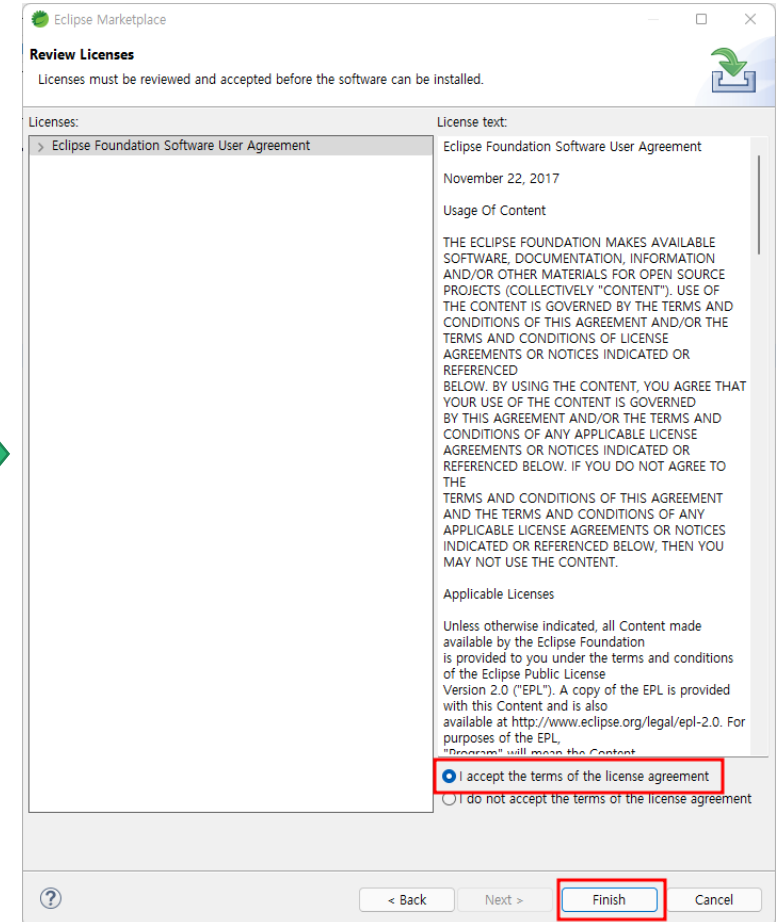
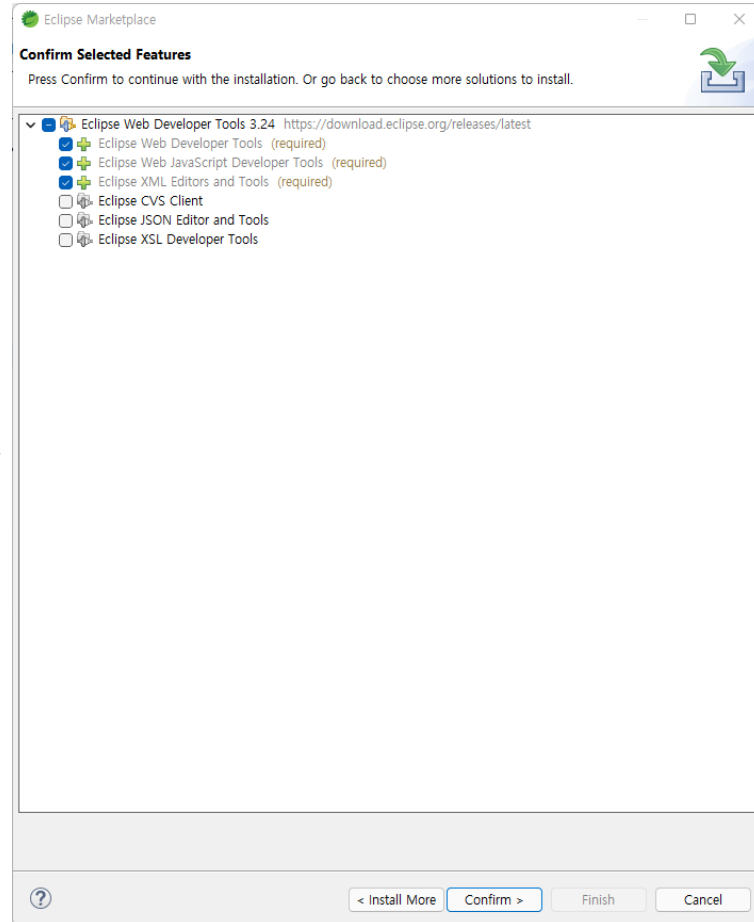
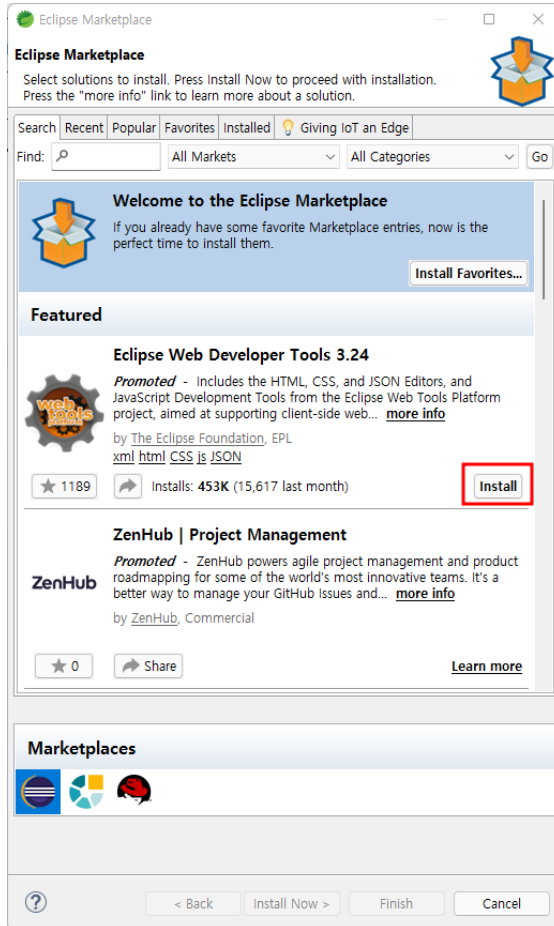
• 브라우저 확인

- 서버 구동이 확인되면 브라우저를 실행시켜 **"localhost:8080"**을 입력, 아래의 화면이 실행결과로 나타나면 설정이 완료된 것이다.
- (클라이언트측 화면을 작성하지 않았기 때문에 아래와 같이 기본 설정된 에러 페이지 출력.)



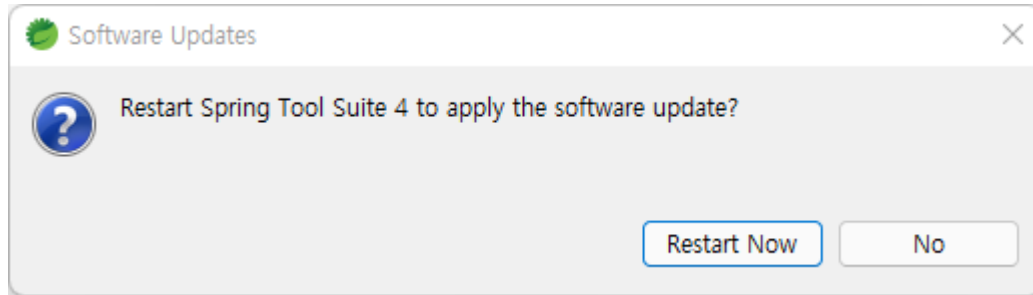
□ HTML 편집을 위한 추가 기능 설치 (1)

- STS 4.9 이후 버전에서는 html, css, javascript를 편집하기 위한 기능을 따로 제공하지 않으므로 추가 설치가 필요
- [Help] - [Eclipse Marketplace] - “Eclipse Web Developer Tools” 검색 후 설치



HTML 편집을 위한 추가 기능 설치 (2)

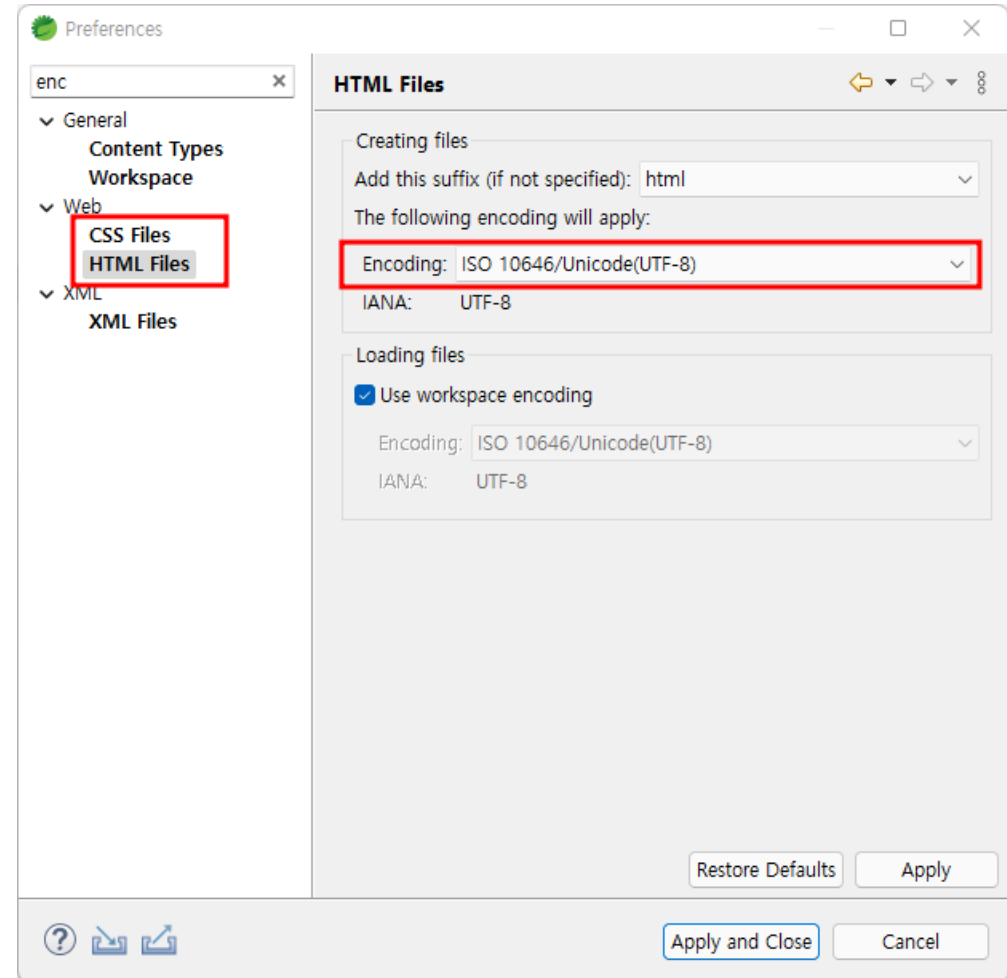
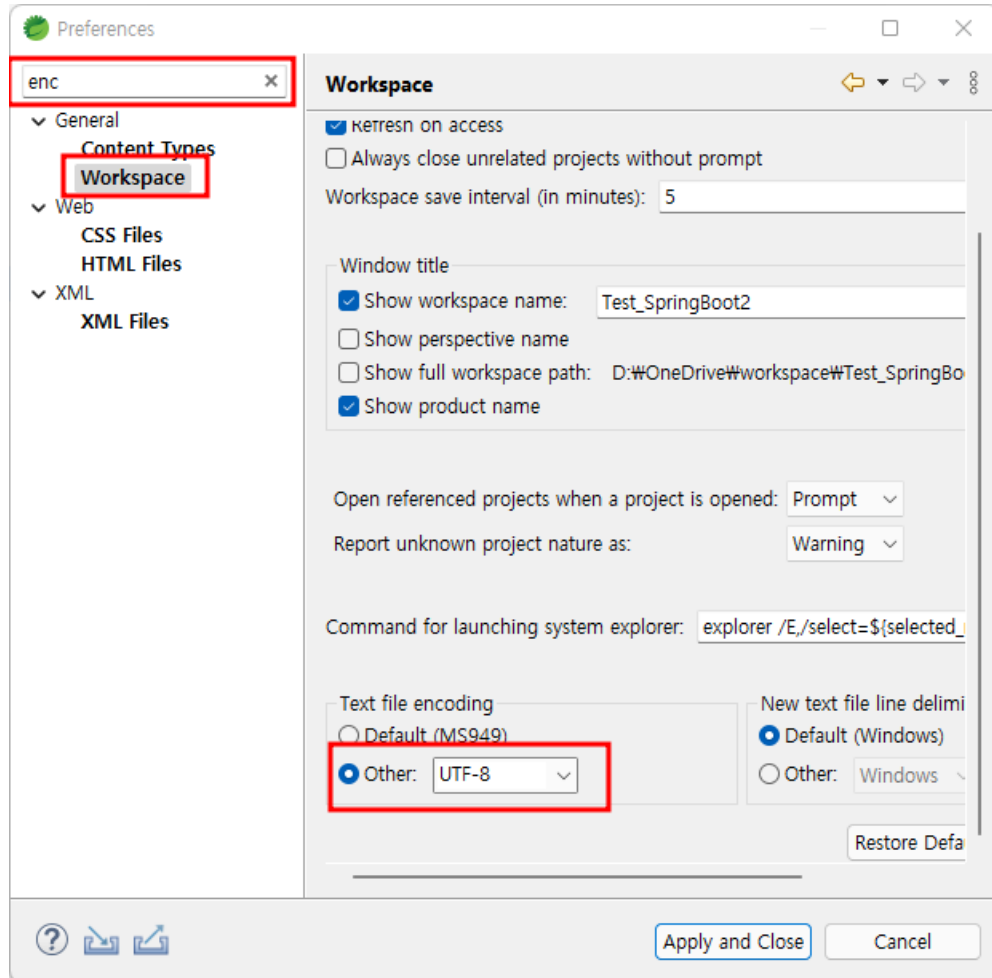
- 설치가 완료되면 STS 재 구동



□ UTF-8 Encoding 설정

- 주요 파일을 생성하기 전 encoding을 utf-8로 설정

- [Window]-[Preferences] 메뉴를 실행시키고 “encoding” 입력하여 Workspace, CSS Files, HTML Files의 설정 수정



[실습] Controller, html 문서의 생성 (1)

- IndexController.java

```
package net.softsociety.web.controller;

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

@Controller
public class IndexController {

    @GetMapping({ "", "/" })
    public String index() {
        return "index";
    }
}
```

[실습] Controller, html 문서의 생성 (2)

- Thymeleaf 사용을 위한 name space의 정의

```
<html xmlns:th="http://www.thymeleaf.org">
```

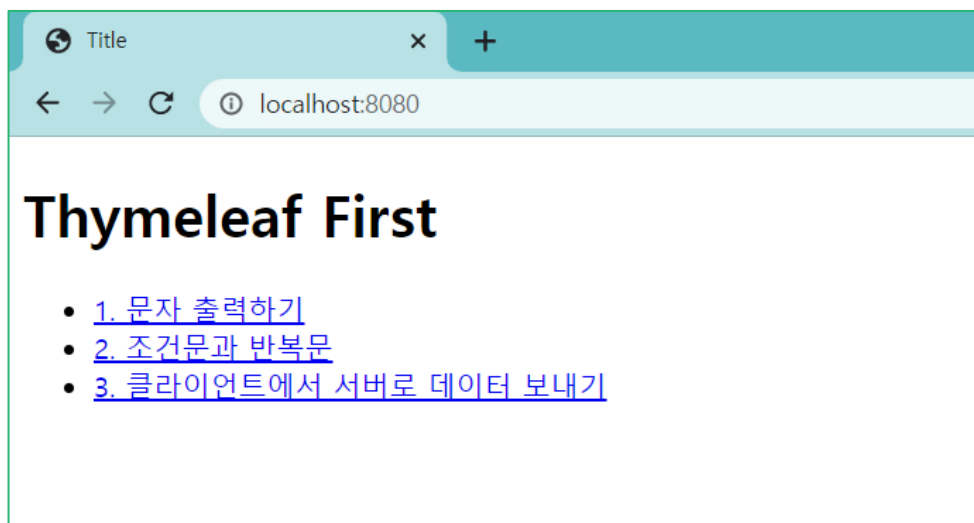
- templatesWindex.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8">
<title>Title</title>
</head>
<body>
  <h1>Thymeleaf First</h1>
  <ul>
    <li><a href="./display/text.html" th:href="@{'/display/text'}">1. 문자 출력하기</a></li>
    <li><a href="./display/condition.html" th:href="@{'/display/condition'}">2. 조건문과 반복문</a></li>
    <li><a href="./display/text.html" th:href="@{'/display/sendData'}">3. 클라이언트에서 서버로 데이터 보내기</a></li>
  </ul>
</body>
</html>
```

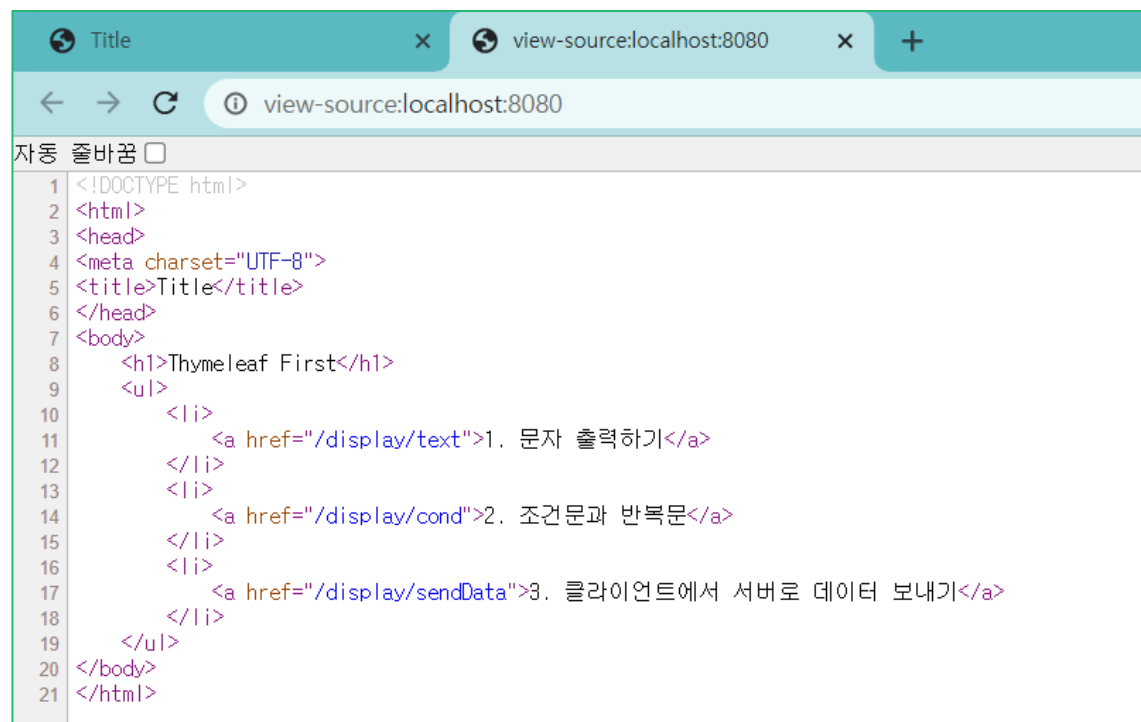
[실습] Controller, html 문서의 생성 (3)

• 서버 구동하여 결과 확인

- IndexController.java, index.html 문서를 작성한 후 브라우저를 이용하여 접속하면 아래와 같은 화면이 출력
- 출력된 화면의 소스를 확인하면 “th:” 로 시작된 템플릿 코드가 html 속성값으로 바뀐 것을 확인할 수 있다.



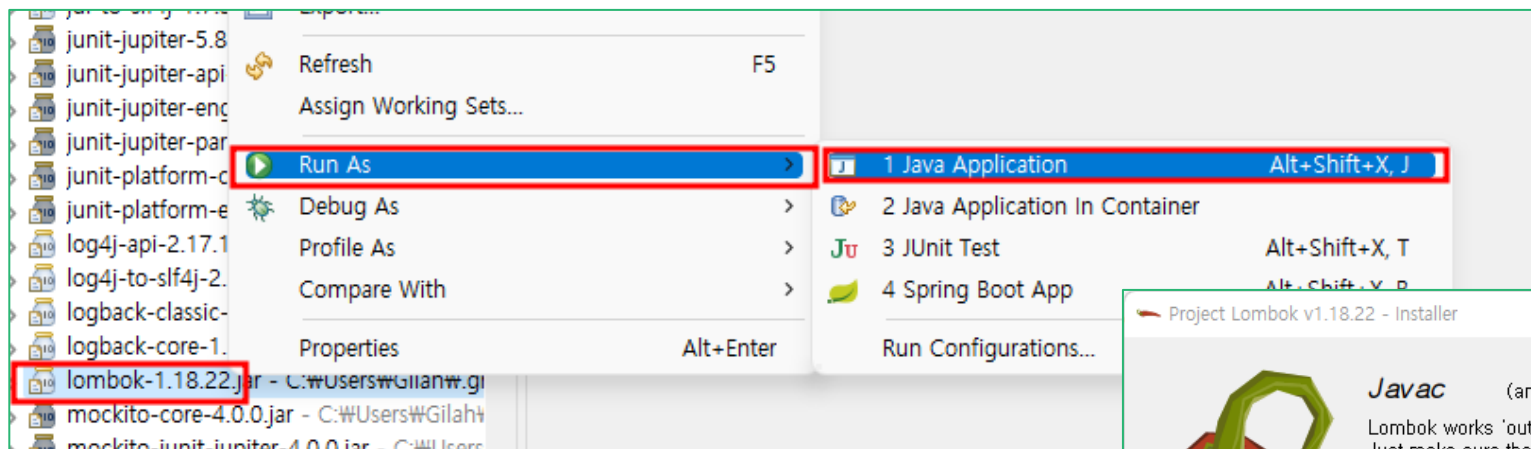
결과



페이지 소스보기

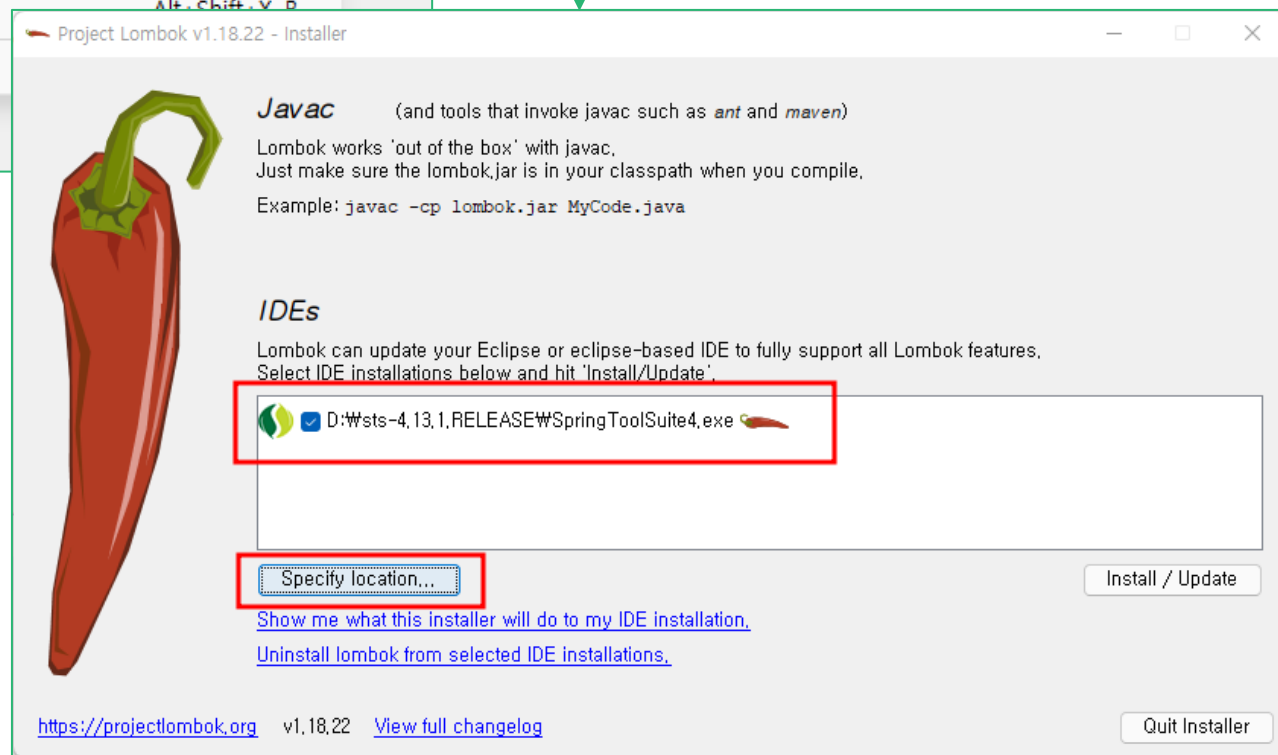
[참고] Lombok 메소드 생성이 안될 때 (1)

- Lombok에서 메소드 생성이 안될 때



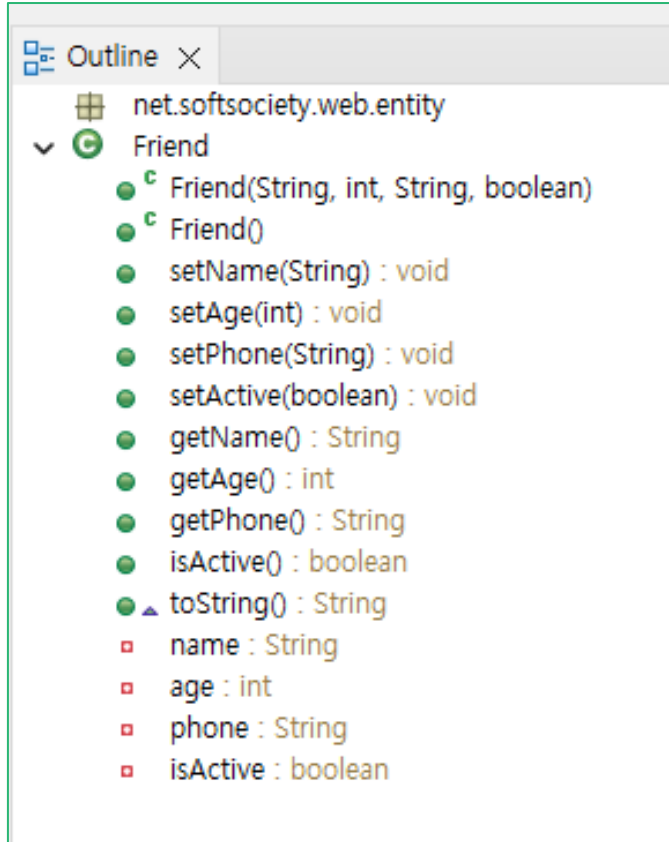
- 1) Package Explorer 창에서 Project and External Dependencies 오픈
- 2) lombok.jar 파일을 찾아 [Run AS]-[Java Application] 명령으로 구동

- 3) STS와 lombok 연동
- 4) 완료 후 STS 재구동



[참고] Lombok 메소드 생성이 안될 때 (2)

- Lombok에서 메소드 생성이 안될 때



5) 필요한 메소드가 생성되는지 outline창에서 확인

▣ Thymeleaf 표준 표현식의 사용

• Text

변수 표현식	<code>\${...}</code>
선택 변수 표현식 / Object 속성값 표현	<code>*{...}</code>
메시지 표현	<code>#{...}</code>
Link URL 표현식	<code>@{...}</code>
Fragment 표현식	<code>~{...}</code>

• Literal 사용

- Thymeleaf는 다양한 종류의 숫자, 혹은 문자 리터럴을 사용할 수 있다.

• 반복문

- `th:each` 를 이용하여 특정 객체의 수 만큼 반복가능

• 조건문

- `th:text` 내에서 조건 연산자를 이용하거나 `th:if`, `th:unless`를 사용하여 조건에 따라 다른 결과물을 출력할 수 있다.

□ 표현의 종류

- Simple expressions:
 - Variable Expressions: `${...}`
 - Selection Variable Expressions: `*{...}`
 - Message Expressions: `#{...}`
 - Link URL Expressions: `@{...}`
 - Fragment Expressions: `~{...}`
- Literals
 - Text literals: `'one text'`, `'Another one!'`,...
 - Number literals: `0`, `34`, `3.0`, `12.3`,...
 - Boolean literals: `true`, `false`
 - Null literal: `null`
 - Literal tokens: `one`, `sometext`, `main`,...
- Text operations:
 - String concatenation: `+`
 - Literal substitutions: `|The name is ${name}|`
- Arithmetic operations:
 - Binary operators: `+`, `-`, `*`, `/`, `%`
 - Minus sign (unary operator): `-`
- Boolean operations:
 - Binary operators: `and` , `or`
 - Boolean negation (unary operator): `!` , `not`
- Comparisons and equality:
 - Comparators: `>` , `<` , `>=` , `<=` (`gt` , `lt` , `ge` , `le`)
 - Equality operators: `==` , `!=` (`eq` , `ne`)
- Conditional operators:
 - If-then: `(if) ? (then)`
 - If-then-else: `(if) ? (then) : (else)`
 - Default: `(value) ?: (defaultvalue)`

[실습] 변수 표현식의 사용 (1) - Entity

- Friend.java

```
package net.softsociety.web.entity;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import lombok.ToString;

@AllArgsConstructor          // 전달인자 있는 생성자
@NoArgsConstructor         // 전달인자 없는 기본생성자
@Setter @Getter
@ToString
public class Friend {
    private String name;
    private int age;
    private String phone;
    private boolean isActive;    // true(활동적), false(정적)
}
```

[실습] 변수 표현식의 사용 (2) - Controller

- TextController.java

```
@Controller
@RequestMapping("/display")
public class TextController {

    /**
     * 요청 "/display/text" 을 처리하는 메소드
     * @return
     */
    @GetMapping("/text")
    public String text(Model model) {
        String korean    = "대한민국 ~ ♥♥♥";
        String english   = "United State America";
        String tag1      = "<b>동해물</b>과 <i>백두산</i>이 마르고
        닳도록";
        String nullData  = null;
        String emptyData = "";

        int number = 1239;
        double pi = Math.PI;

        Friend friend    = new Friend("저팔계", 23, "010-2222-3333", true);
        Friend friend1   = new Friend("전우치", 33, "010-2222-4444", false);

        model.addAttribute("korean", korean );
        model.addAttribute("english", english );
        model.addAttribute("tag1", tag1 );
        model.addAttribute("nullData", nullData );
        model.addAttribute("emptyData", emptyData );
        model.addAttribute("number", number );
        model.addAttribute("pi", pi );
        model.addAttribute("friend", friend);
        model.addAttribute("friend1", friend1);

        return "display/text";
    }
}
```

[실습] 변수 표현식의 사용 (3) - html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>글 출력</title>
</head>
<body>
  <div class="container">
    <div>
      <a href="../index.html" th:href="@{/}">
        
      </a>
    </div>
    <h1>각종 데이터의 출력</h1>
    <h2>(1) 일반 데이터</h2>
    <ul>
      <li>한글 : <span th:text="${korean}">한글</span></li>
      <li>영어 : <span th:text="${english}">영어</span></li>
      <li>태그 인식 불가: <span th:text="${tag1}">태그1</span></li>
      <li>태그 인식 : <span th:utext="${tag1}">태그2</span></li>
      <li>널문자 : <span th:text="${nullData}">널문자</span></li>
      <li>빈문자 : <span th:text="${emptyData}">빈문자</span></li>
      <li>정수 : <span th:text="${number}">정수</span></li>
      <li>실수 : <span th:text="${pi}">실수</span></li>
    </ul>
```

[실습] 변수 표현식의 사용 (4) - html

```
<h2>(2) 객체</h2>
<ul>
  <li>객체 출력-1: <span th:text="{friend}">저팔계</span></li>
  <li>객체 출력-2: <span th:text="{friend.name}">저팔계</span></li>
  <li>객체 출력-3
    <ul th:object="{friend1}">
      <li>이름 : <span th:text="{name}">전우치</span></li>
      <li>나이 : <span th:text="{age}">나이</span></li>
      <li>전화번호 : <span th:text="{phone}">전화번호</span></li>
      <li>성향 : <span th:text="{isActive}">성향</span></li>
    </ul>
  </li>
  <li>인라인 출력-1 : <span>[{friend.name}]</span></li>
  <li>인라인 출력-2 : <span>[{friend}]</span></li>
</ul>
</body>
</html>
```


[실습] 변수 표현식의 사용 (4) - 결과



각종 데이터의 출력

(1) 일반 데이터

- 한글 : 대한민국 ~ ♥♥♥
- 영어 : United State America
- 태그 인식 불가: 동해물과 <i>백두산</i>이 마르고 닳도록
- 태그 인식 : 동해물과 백두산이 마르고 닳도록
- 널문자 :
- 빈문자 :
- 정수 : 1239
- 실수 : 3.141592653589793

(2) 객체

- 객체 출력-1: Friend(name=저팔계, age=23, phone=010-2222-3333, isActive=true)
- 객체 출력-2: 저팔계
- 객체 출력-3
 - 이름 : 전우치
 - 나이 : 33
 - 전화번호 : 010-2222-4444
 - 성향 : false
- 인라인 출력-1 : 저팔계
- 인라인 출력-2 : Friend(name=저팔계, age=23, phone=010-2222-3333, isActive=true)

▣ th:text (1)

- th:text
 - 서버에서 저장된 텍스트를 출력
 - 출력데이터 내에 tag가 포함되어 있어도 text로 출력
 - null이나 빈문자는 표현하지 않으며, 숫자 또는 날짜도 텍스트로 출력
 - `${...}`와 사용

[예]

한글 : <code>한글</code>	한글 : 대한민국 ~ ♥♥♥
태그 : <code>태그1</code>	태그 : <code>동해물과 <i>백두산</i>이 마르고 닳도록</code>
널문자 : <code>널문자</code> 빈문자 : <code>빈문자</code>	널문자 : 빈문자 :
정수 : <code>정수</code> 실수 : <code>실수</code>	정수 : 1239 실수 : 3.141592653589793

▣ th:text (2)

- **th:text**
 - 3.0 버전 이상에서는 인라인 표현방식 `[[...]]` 을 이용하여 출력가능
 - 객체의 경우 `toString()`이나 멤버변수의 `getter()` 값을 출력

[예]

<code>객체 출력-1: 저팔계</code>	객체 출력-1: Friend(name=저팔계, age=23, phone=010-2222-3333, isActive=true)
<code>객체 출력-2: 저팔계</code>	객체 출력-2: 저팔계
<code>인라인 출력-1 : [[{friend.name}]]</code>	인라인 출력-1 : 저팔계
<code>인라인 출력-2 : [[{friend}]]</code>	인라인 출력-2 : Friend(name=저팔계, age=23, phone=010-2222-3333, isActive=true)

▣ th:object

- th:object

- 서버에서 저장된 객체값을 출력
- `${...}`, `*{...}` 와 사용

[예]

<pre>이름 : 전우치 나이 : 나이 전화번호 : 전화번호 성향 : 성향</pre>	이름 : 전우치 나이 : 33 전화번호 : 010-2222-4444 성향 : false
<pre>객체 출력-3 <ul th:object="\${friend1}"> 이름 : 전우치 나이 : 나이 전화번호 : 전화번호 성향 : 성향 </pre>	이름 : 전우치 나이 : 33 전화번호 : 010-2222-4444 성향 : false

▣ th:text / th:utext

- th:utext

- 서버에서 저장된 텍스트에 tag가 포함되어 있으면 tag를 rendering 하여 출력

[예]

<code>태그 인식 불가: 태그1</code>	태그 인식 불가: 동해물과 <i>백두산</i>이 마르고 닳도록
<code>태그 인식 : 태그2</code>	태그 인식 : 동해물과 백두산이 마르고 닳도록

▣ URL Link 표현

• html의 URL 관련 attribute

- , <a>, <form>... 태그의 attribute와 동일
- th:href, th:src, th:action 등 외부 URL을 사용하는 태그의 attribute로 사용
- @{...} 과 함께 사용
- Thymeleaf는 특정 HTML5 속성과 동일한 속성이 많다.
- 참고 : <https://www.thymeleaf.org/doc/tutorials/3.0/usingthymeleaf.html>

[예]

<code> </code>	/ 요청 이미지 URL
<code><a th:href="@{/display/sendOne(username=Gildong, age=23)}"></code>	쿼리 String 전송. 아래 표현과 동일 localhost:8080/display/sendOne?username=Gildong&age=23
<code><form th:action="@{/display/sendVO}" method="POST"></code>	/display/sendVO 요청 URL로 form data 전송

▣ 기본 객체의 표현

내용	설명	예
#dates	java.util.Date 객체를 위한 메소드	
#calendars	java.util.Calendar 객체를 위한 메소드	
#numbers	숫자 객체의 서식을 지정	
#strings	startsWith, prepending/appending를 포함한 String 객체에 대한 메소드	
#lists	list 데이터 관련 메소드	
#maps	map 데이터 관련 메소드	HashMap
#HttpServletRequest	http 프로토콜의 request 정보를 서블릿에게 전달하기 위해 사용되며 헤더정보, 파라미터, 쿠키, URI 등이 정보를 읽어들이는 메소드 포함	URL

[실습] 여러 종류의 객체 표현(1) - Java

```
@GetMapping("/basicObj")
public String basicObj(Model model) {
    int intNum = 1234560;
    double dblNum = 1234.5678;
    double percent = 0.0325;
    double money = 568000;

    Date date = new Date();
    Calendar calendar = Calendar.getInstance();
    String korean = "동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리나라 만세";
    String english = "I have a dream, a song to sing. To help me cope with anything";

    boolean bool = true;

    model.addAttribute("intNum", intNum);
    model.addAttribute("dblNum", dblNum);
    model.addAttribute("percent", percent);
    model.addAttribute("money", money);

    model.addAttribute("date", date);
    model.addAttribute("calendar", calendar);

    model.addAttribute("date", date);
    model.addAttribute("calendar", calendar);
}
```

[실습] 여러 종류의 객체 표현(2) - Java

```
model.addAttribute("korean", korean);
model.addAttribute("english", english);

model.addAttribute("bool", bool);

List<String> list = new ArrayList<>();
list.add("사과");
list.add("배");
list.add("오렌지");
list.add("바나나");

Map<String, Friend> map = new HashMap<>();
map.put("son", new Friend("손오공", 34, "101-1111-2222", true));
map.put("ojung", new Friend("사오정", 24, "101-2222-2222", false));
map.put("palgae", new Friend("저팔계", 29, "101-4444-2222", true));

model.addAttribute("list", list);
model.addAttribute("map", map);

return "display/basicObj";
}
```

[실습] 여러 종류의 객체 표현(3) - HTML

```
<body>
  <div class="container">
    <div>
      <a href="../index.html" th:href="@{}/">
        
      </a>
    </div>

    <h1>Basic Object의 표현</h1>
    <h2>(4) 객체의 기본 출력</h2>
    <ul>
      <li>Integer : <span th:text="${intNum}">Integer</span></li>
      <li>Double : <span th:text="${dblNum}">Double</span></li>
      <li>Date : <span th:text="${date}">Date</span></li>
      <li>Calendar : <span th:text="${calendar}">Calendar</span></li>
      <li>String : <span th:text="${korean}">String</span></li>
      <li>Boolean : <span th:text="${bool}">Boolean</span></li>
      <li>ArrayList : <span th:text="${list}">ArrayList</span></li>
      <li>HashMap : <span th:text="${map}">HashMap</span></li>
    </ul>
  </div>
```

(4) 객체의 기본 출력

- Integer : 1234560
- Double : 1234.5678
- Date : Mon Mar 14 14:36:50 KST 2022
- Calendar :
java.util.GregorianCalendar[time=1647236210590,areFieldsSet=true,areAllFieldsSet=
- String : 동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리나라 만세
- Boolean : true
- ArrayList : [사과, 배, 오렌지, 바나나]
- HashMap : {son=Friend(name=손오공, age=34, phone=101-1111-2222, active=true), pal
- 사오정, age=24, phone=101-2222-2222, active=false)}

[실습] 여러 종류의 객체 표현(4) - HTML

<h2>(5) 객체 Method(Date)</h2>

Date Format :

day :

month :

monthName :

monthNameShort :

year :

dayOfWeek :

dayOfWeekName :

dayOfWeekNameShort :

hour :

minute :

second :

millisecond :

(5) 객체 Method(Date)

- Date Format : 14/03/2022 14:36
- day : 14
- month : 3
- monthName : 3월
- monthNameShort : 3월
- year : 2022
- dayOfWeek : 2
- dayOfWeekName : 월요일
- dayOfWeekNameShort : 월
- hour : 14
- minute : 36
- second : 50
- millisecond : 589

[실습] 여러 종류의 객체 표현(5) - HTML

<h2>(6) 객체 Method(Calendar) : Date 객체와 동일</h2>

Calendar Format :

day :

month :

monthName :

monthNameShort :

year :

dayOfWeek :

dayOfWeekName :

dayOfWeekNameShort :

hour :

minute :

second :

millisecond :

(6) 객체 Method(Calendar) : Date 객체와 동일

- Calendar Format : 14/03/2022 14:36
- day : 14
- month : 3
- monthName : 3월
- monthNameShort : 3월
- year : 2022
- dayOfWeek : 2
- dayOfWeekName : 월요일
- dayOfWeekNameShort : 월
- hour : 14
- minute : 36
- second : 50
- millisecond : 590

[실습] 여러 종류의 객체 표현(6) - HTML

<h2>(7) 객체 Method(숫자): 최소 정수 자릿수 및 천 단위 구분 기호(POINT, COMMA, WHITESPACE, NONE, DEFAULT)</h2>


```
<li>정수 10자리: <span th:text="{#numbers.formatInteger(intNum, 10)}"></span></li> <!-- 10자리로 맞추 -->
<li>천 단위 POINT : <span th:text="{#numbers.formatInteger(intNum, 3, 'POINT')}"></span></li> <!-- 3자리마다 점 -->
<li>천 단위 COMMA : <span th:text="{#numbers.formatInteger(intNum, 3, 'COMMA')}"></span></li> <!-- 3자리마다 콤마 -->
<li>천 단위 공백 : <span th:text="{#numbers.formatInteger(intNum, 3, 'WHITESPACE')}"></span></li> <!-- 3자리마다 공백 -->
<li>천 단위 NONE : <span th:text="{#numbers.formatInteger(intNum, 3, 'NONE')}"></span></li> <!-- 3자리마다 None -->
<li>천 단위 DEFAULT : <span th:text="{#numbers.formatInteger(intNum, 3)}"></span></li> <!-- Default -->
<li>-----</li>
<li>Double : <span th:text="{dblNum}">Double</span></li>
<li>소수점 2자리: <span th:text="{#numbers.formatDecimal(dblNum, 3, 2)}"></span></li>
<li>천 단위 COMMA, 소수점 2자리 POINT: <span th:text="{#numbers.formatDecimal(dblNum,3,'COMMA', 2,'POINT')}"></span></li>
<li>천 단위 공백, 소수점 3자리 DEFAULT: <span th:text="{#numbers.formatDecimal(dblNum, 3,'WHITESPACE', 3, 'DEFAULT')}"></span></li>
<li>-----</li>
<li>백분율(소수이하 2자리) : <span th:text="{#numbers.formatPercent(percent, 0, 2)}"></span></li> <!-- 3개의 인자 사용해야함 -->
<li>Currency : <span th:text="{#numbers.formatCurrency(money)}"></span></li>
```


(7) 객체 Method(숫자): 최소 정수 자릿수 및 천 단위 구분 기호

- 정수 10자리: 0001234560
- 천 단위 POINT : 1.234.560
- 천 단위 COMMA : 1,234,560
- 천 단위 공백 : 1 234 560
- 천 단위 NONE : 1234560
- 천 단위 DEFAULT : 1234560
- -----
- Double : 1234.5678
- 소수점 2자리: 1234.57
- 천 단위 COMMA, 소수점 2자리 POINT: 1,234.57
- 천 단위 공백, 소수점 3자리 DEFAULT: 1 234.568
- -----
- 백분율(소수이하 2자리) : 3.25%
- Currency : ₩568,000

[실습] 여러 종류의 객체 표현(7) - HTML

<h2>(8) 문자열 함수의 사용</h2>

한글:

문자열의 길이:

문자열 연결:

영어:

isEmpty:

문자열 포함:

startsWith:

endsWith:

문자열 시작위치: <!-- 찾는 문자열이 없으면 -1 -->

특정 위치의 부분 문자열:

특정 위치 뒤 부분 문자열:

특정 위치 앞 부분 문자열:

치환:

(8) 문자열 함수의 사용

- 한글: 동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리나라 만세
- 문자열의 길이: 35
- 문자열 연결: 동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리나라 만세~I have a dream
- 영어: I have a dream, a song to sing. To help me cope with anything
- isEmpty: false
- 문자열 포함: false
- startsWith: false
- endsWith: true
- 문자열 시작위치: 5
- 특정 위치의 부분 문자열: dream
- 특정 위치 뒤 부분 문자열: 두산이 마르고 닳도록 하느님이 보우하사 우리나라 만세
- 특정 위치 앞 부분 문자열: I have a dream, a song to sing.
- 치환: 동해물과 백두산이 마르고 닳도록 하느님이 보우하사 대한민국 만세

[실습] 여러 종류의 객체 표현(8) - HTML

<h2>(9) ArrayList, HashMap</h2>

ArrayList : ArrayList
ArrayList size:
ArrayList size(위와 동일) :
isEmpty:
isEmpty(위와 동일):
List를 문자열로 Join:
문자열을 List로 Split:
데이터 포함 여부:

HashMap containsKey: HashMap
HashMap isEmpty: HashMap
HashMap size: HashMap
HashMap size: HashMap

(9) ArrayList, HashMap

- ArrayList : [사과, 배, 오렌지, 바나나]
- ArrayList size: 4
- ArrayList size(위와 동일) : 4
- isEmpty: false
- isEmpty(위와 동일): false
- List를 문자열로 Join: 사과,배,오렌지,바나나
- 문자열을 List로 Split: [동해물과 백두산이 마르고 닳도록 하느님이 보우하사 우리나라 만세]
- 데이터 포함 여부: true
- -----
- HashMap containsKey: true
- HashMap isEmpty: false
- HashMap size: 3
- HashMap size: 3

[실습] 여러 종류의 객체 표현(9) - HTML

```
<div>
<h2>(10) HttpServletRequest 객체</h2>
<ul>
  <li>requestURL : [[ ${#HttpServletRequest.requestURL} ]]</li>
  <li>serverPort : [[ ${#HttpServletRequest.serverPort} ]]</li>
  <li>serverName : [[ ${#HttpServletRequest.serverName} ]]</li>
  <li>servletPath : [[ ${#HttpServletRequest.servletPath} ]]</li>
  <li>remoteAddr : [[ ${#HttpServletRequest.remoteAddr} ]]</li>
  <li>remoteAddr : [[ ${#HttpServletRequest.remoteAddr} ]]</li>
  <li>method : [[ ${#HttpServletRequest.method} ]]</li>
  <li>requestedSessionId : [[ ${#HttpServletRequest.requestedSessionId} ]]</li>
  <li>scheme : [[ ${#HttpServletRequest.scheme} ]]</li>
  <li>protocol : [[ ${#HttpServletRequest.protocol} ]]</li>
  <li>locale : [[ ${#HttpServletRequest.locale} ]]</li>
  <li>secure : [[ ${#HttpServletRequest.secure} ]]</li>
</ul>
</div>
```

(10) HttpServletRequest 객체

- requestURL : http://localhost:8882/test4/
- serverPort : 8882
- serverName : localhost
- servletPath : /
- remoteAddr : 0:0:0:0:0:0:0:1
- remoteAddr : 0:0:0:0:0:0:0:1
- method : GET
- requestedSessionId :
- scheme : http
- protocol : HTTP/1.1
- locale : ko_KR
- secure : false

□ 반복문(1)

내용	설명	예
th:each	th:each 속성을 이용하여 값을 반복. - Token 형태 문자열을 분해하여 반복 - ArrayList 내의 데이터를 반복	<pre> <!-- 문자열을 분해(split)하여 data변수에 넣는다. --> <ul th:with="data=\${#strings.listSplit(fruit, ', ')}"> <li th:each="fruit : \${data}"> </pre>
		<pre> <li th:each="str : \${strList}"> 과일 </pre>
		<pre> <tr th:each="friend: \${friendList}"> <td th:text="\${friend.name}">이름</td> <td th:text="\${friend.age}">나이</td> <td th:text="\${friend.phone}">전화번호</td> <td th:text="\${friend.active} ? '활동적' : '정적'">성향</td> </tr> </pre>

□ 반복문(2)

내용	설명	예
th:each	<p>HashMap 데이터 반복</p> <ul style="list-style-type: none"> - Key, Value 데이터를 출력 - th:if속성의 연산 값에 따라 값을 출력하고, th:class의 속성을 적용할 수 있다. - th:unless는 not if와 동일한 의미 	<pre><tr th:each="friend: \${friendMap}"> <td th:text="\${friend.key}">이름</td> <td th:text="\${friend.value.name}">이름</td> <td th:text="\${friend.value.age}">나이</td> <td th:text="\${friend.value.phone}">전화번호</td> <td th:text="\${friend.value.active} ? '활동적' : '정적'">성향</td> </tr></pre>
		<pre><li th:each="num, status : \${numList}"> ----- </pre>

[실습] 반복문(1) - Controller

```
@Controller
@RequestMapping("/display")
public class ConditionController {

    @GetMapping("/condition")
    public String condition(Model model) {
        String fruit = "Apple, orange, Mango, Banana, Strawberry";

        List<String> strList = new ArrayList<>();

        strList.add("사과");
        strList.add("배");
        strList.add("오렌지");
        strList.add("바나나");

        List<Friend> friendList = new ArrayList<>();
        friendList.add(new Friend("손오공", 34, "101-1111-2222", true));
        friendList.add(new Friend("사오정", 24, "101-2222-2222", false));
        friendList.add(new Friend("저팔계", 31, "101-3333-2222", false));
        friendList.add(new Friend("전우치", 29, "101-4444-2222", true));

        Map<String, Friend> friendMap = new HashMap<>();
        friendMap.put("son", new Friend("손오공", 34, "101-1111-2222", true));
        friendMap.put("ojung", new Friend("사오정", 24, "101-2222-2222", false));
        friendMap.put("palgae", new Friend("저팔계", 29, "101-4444-2222", true));
```

```
        List<String> numList = new ArrayList<>();

        for(int i=0; i<50; ++i)
            numList.add("Current Number="+i);

        model.addAttribute("fruit", fruit);
        model.addAttribute("strList", strList);
        model.addAttribute("friendList", friendList);
        model.addAttribute("friendMap", friendMap);
        model.addAttribute("numList", numList);

        return "display/condition";
    }
}
```

[실습] 반복문(2) - HTML

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>조건문과 반복문 출력</title>
</head>
<body>
<div class="container">
  <div>
    <a href="../index.html" th:href="@{/}">
      
    </a>
  </div>

  <h1>반복문 사용</h1>
  <h2>(10) 문자열 분해하여 반복</h2>
  <ul th:with="data=${#strings.listSplit(fruit, ', ')}"> <!-- 문자열을 분해(list)하여 data변수에 넣는다. -->
    <li th:each="fruit : ${data}">
      <span th:text="${fruit}"></span>
    </li>
  </ul>
</div>
```



반복문 사용

(10) 문자열 분해하여 반복

- Apple
- orange
- Mango
- Banana
- Strawberry

[실습] 반복문(3) - HTML

<h2>(11) ArrayList 반복</h2>

<li th:each="str : \${strList}">

과일

<h2>(12) ArrayList에 들어있는 VO 데이터 출력</h2>

<p>저장된 친구 수 : </p>

<table border="1">

<tr>

<th>이름</th>

<th>나이</th>

<th>전화번호</th>

<th>성향</th>

</tr>

<tr th:each="friend: \${friendList}">

<td th:text="\${friend.name}">이름</td>

<td th:text="\${friend.age}">나이</td>

<td th:text="\${friend.phone}">전화번호</td>

<td th:text="\${friend.active} ? '활동적' : '정적'">성향</td>

</tr>

</table>

(11) ArrayList 반복

- 사과
- 배
- 오렌지
- 바나나

(12) ArrayList에 들어있는 vo 데이터 출력

저장된 친구 수 : 4

이름	나이	전화번호	성향
손오공	34	101-1111-2222	활동적
사오정	24	101-2222-2222	정적
저팔계	31	101-3333-2222	정적
전우치	29	101-4444-2222	활동적

[실습] 반복문(4) - HTML

<h2>(13) HashMap에 들어있는 VO 데이터 1개 출력</h2>

<p th:text="\${friendMap['son']}"></p>

<h2>(14) HashMap에 들어있는 모든 VO 데이터 출력</h2>

<p>저장된 친구 수 : </p>

<table border="1">

<tr>

<th>Key</th>

<th>이름</th>

<th>나이</th>

<th>전화번호</th>

<th>성향</th>

</tr>

<tr th:each="friend: \${friendMap}">

<td th:text="\${friend.key}">이름</td>

<td th:text="\${friend.value.name}">이름</td>

<td th:text="\${friend.value.age}">나이</td>

<td th:text="\${friend.value.phone}">전화번호</td>

<td th:text="\${friend.value.active} ? '활동적' : '정적'">성향</td>

</tr>

</table>

(13) HashMap에 들어있는 vo 데이터 1개 출력

Friend(name=손오공, age=34, phone=101-1111-2222, active=true)

(14) HashMap에 들어있는 모든 vo 데이터 출력

저장된 친구 수 : 3

Key	이름	나이	전화번호	성향
son	손오공	34	101-1111-2222	활동적
palgae	저팔계	29	101-4444-2222	활동적
ojung	사오정	24	101-2222-2222	정적

[실습] 반복문(5) - HTML

```
<br>
<h2>(15) List와 상태값을 함께 출력</h2>
<ul>
  <li th:each="num, status : ${numList}">
    <span th:text="${status.index}"></span>
    <span th:if="${status.index % 5 == 0}" th:text="${num}" th:class="'mystyle'"></span>
    <span th:unless="${status.index % 5 == 0}">-----</span>
  </li>
</ul>
</div>
</body>
</html>
```

(15) List와 상태값을 함께 출력

- 0 Current Number=0
- 1 -----
- 2 -----
- 3 -----
- 4 -----
- 5 Current Number=5
- 6 -----
- 7 -----
- 8 -----
- 9 -----
- 10 Current Number=10
- 11 -----
- 12 -----
- 13 -----
- 14 -----

□ 연산자

내용	설명	예
산술연산	+, -, *, / , %	일반적인 산술연산 기능 수행 값 + 10 : <code></code> 값 - 10 : <code></code> 값 * 3: <code></code> 값 / 3: <code></code> 값 % 2: <code></code>
문자열 결합	+, (필터)	문자열을 결합하거나 두 개의 기호 사이에 문자열을 넣으면 그대로 출력 <code></code> <code> ' + \${two}, \${three}, 너구리, 오징어 "></code>
비교연산자	>(gt), <(lt), >=(ge), <=(le), ==(eq), !=(ne)	수치 데이터의 경우 기호나 문자를 이용하여 연산 가능 비교연산 ==(eq): <code></code> 논리연산 (and): <code> 10 and \${intNum} < 30"></code>
논리연산자	and, or, not(!)	둘 이상의 비교연산자를 논리적으로 연산을 하거나 현재 논리식을 부정할 때 사용 <code><span th:text="not (\${intNum} < 10)"></code>
조건연산자	? :	th:text를 이용하여 조건식으로 사용할 수 있으며 th:if와 동일하다. <code></code>

[실습] 다양한 연산자 (1) - Controller

```
@Controller
@RequestMapping("/display")
public class Operator {
    @GetMapping("/operator")
    public String text(Model model) {
        String name = "James Dean";
        String one = "한놈";
        String two = "두시기";
        String three = "석삼";

        Friend friend = new Friend("저팔계", 23, "010-2222-3333", false);

        int intNum = 22;
        double dblNum = 42.195;

        model.addAttribute("name", name);
        model.addAttribute("one", one);
        model.addAttribute("two", two);
        model.addAttribute("three", three);
        model.addAttribute("friend", friend);
        model.addAttribute("intNum", intNum);
        model.addAttribute("dblNum", dblNum );
        return "display/operator";
    }
}
```

[실습] 다양한 연산자 (2) - HTML

```
<body>
<div class="container">
  <div>
    <a href="../index.html" th:href="@{/}">
      
    </a>
  </div>

  <h1>연산자</h1>
  <h2>(16) 문자열 결합(+ or |)</h2>
  <ul>
    <li>+ 기호로 문자열 연결-1 : <span th:text="'어서오세요, ' + ${friend.name} + '님!'"></span></li>
    <li>| 기호로 문자열 연결-2 : <span th:text="${one} + ' => ' + |${two}, ${three}, 너구리, 오징어|">
      </span></li>
  </ul>

  <h2>(17) 산술 연산 (+ - * / %)</h2>
  <ul>
    <li>값 : <span th:text="${intNum}"></span></li>
    <li>값 + 10 : <span th:text="${intNum} + 10"></span></li>
    <li>값 - 10 : <span th:text="${intNum} - 10"></span></li>
    <li>값 * 3 : <span th:text="${intNum} * 3"></span></li>
    <li>값 / 3 : <span th:text="${intNum} / 3"></span></li>
    <li>값 % 2 : <span th:text="${intNum} % 2"></span></li>
  </ul>
```



연산자

(16) 문자열 결합(+ or |)

- + 기호로 문자열 연결-1 : 어서오세요, 저팔계님!
- | 기호로 문자열 연결-2 : 한놈 => 두시기, 석삼, 너구리, 오징어

(17) 산술 연산 (+ - * / %)

- 값 : 22
- 값 + 10 : 32
- 값 - 10 : 12
- 값 * 3 : 66
- 값 / 3 : 7.3333333333
- 값 % 2 : 0

[실습] 다양한 연산자 (3) - HTML

<h2>(18) 비교 연산 / 논리 연산 / 조건 연산</h2>

값 :

비교연산 ==(eq):

비교연산 !=(ne):

비교연산 > (gt):

비교연산 >=(ge):

비교연산 < (lt):

비교연산 <=(le):

논리연산 (and): 10 and {intNum} < 30">

논리연산 (or): <span th:text="{intNum} < 10 or {intNum} > 30">

논리연산 !(not): <span th:text="not ({intNum} < 10)">

조건연산 (? :):

(18) 비교 연산 / 논리 연산 / 조건 연산

- 값 : 22
- 비교연산 ==(eq): true
- 비교연산 !=(ne): false
- 비교연산 > (gt): false
- 비교연산 >=(ge): true
- 비교연산 < (lt): false
- 비교연산 <=(le): true
- 논리연산 (and): true
- 논리연산 (or): false
- 논리연산 !(not): true
- 조건연산 (? :): 22은 짝수

[실습] 다양한 연산자 (4) - HTML

```
<h2>(19) th:if, th:unless를 이용한 조건식</h2>
<ul>
  <li th:if="{friend.active}==true">
    <span>외향적인 친구</span>
  </li>
  <li th:unless="{friend.active}==true">
    <span>내향적인 성향의 친구</span>
  </li>
  <!-- 위의 th:if와 비교 -->
  <li th:text="{friend.active} ? '활동적' : '정적'">성향</li>
</ul>
</div>
</body>
</html>
```

(19) th:if, th:unless를 이용한 조건식

- 내향적인 성향의 친구
- 정적

▣ 하이퍼링크로 서버에 데이터 전송

• 데이터 전송

- 클라이언트에서 서버로 데이터를 전송하는 방법에는 여러가지가 있는데, a태그를 이용하는 방법

```
<a href="#" th:href="@{/display/sendOne(name=Gildong)}">데이터 전송1</a>
```

```
<!-- localhost/display/sendData?name=Gildong 과 동일한 표현 -->
```

```
<a href="#" th:href="@{/display/sendOne(name=Gildong, age=33)}">데이터 전송2</a>
```

```
<!-- localhost/display/sendData?name=Gildong&age=33 과 동일한 표현 -->
```

```
<a href="#" th:href="@{/display/sendOne}">데이터 전송3</a>
```

```
<!-- localhost/display/sendData 과 동일한 표현 -->
```

▣ Form 태그 내에 입력된 데이터 전송

• 데이터 전송

- Form태그로 데이터를 전송하는 방법

```
<form th:action="@{/display/sendVO}" method="POST">
  <label>이름 : <input type="text" name="name" placeholder="이름을 입력"></label><br>
  <label>나이 : <input type="text" name="age" placeholder="나이를 입력"></label><br>
  <label>전화번호 : <input type="text" name="phone" placeholder="전화번호 입력"></label><br>
  <label>성향 : <input type="radio" name="active" value="1" checked> 활동적
               <input type="radio" name="active" value="0"> 정적
  </label><br>
  <input type="submit" value="전송">
</form>
```

[실습] 데이터 전송 (1) - Controller

```
@Controller
@RequestMapping("/display")
@Slf4j
public class SendDataController {

    @GetMapping("/sendData")
    public String sendData() {

        return "display/sendData";
    }

    @GetMapping("/sendOne")
    public String sendData(Friend friend) {
        log.info("전송된 데이터 = {}", friend.toString());
        return "display/sendData";
    }

    @PostMapping("/sendVO")
    public String sendVO(Friend friend) {
        log.info("전송된 데이터 = {}", friend.toString());
        return "display/sendData";
    }
}
```


[실습] 하이퍼링크로 서버에 데이터 전송 (1) - HTML

```
<body>
<div class="container">
  <div>
    <a href="../index.html" th:href="@{/}">
      
    </a>
  </div>

  <h1>데이터 전송하기</h1>
  <h2>(20) 하이퍼 링크로 하나의 데이터 전송</h2>
  <div>
    <!-- www.naver.com/display/sendData?name=홍길동 -->
    <a href="#" th:href="@{/display/sendOne(name=Gildong)}">데이터 전송1</a>
    <a href="#" th:href="@{/display/sendOne(name=Gildong, age=33)}">데이터 전송2</a>
    <a href="#" th:href="@{/display/sendOne}">데이터 전송3</a>
  </div>
</div>
```

[실습] Form에 입력된 데이터를 서버로 데이터 전송 (2) - HTML

<h2>(21) Form에 입력된 데이터 전송</h2>

```
<form th:action="@{/display/sendVO}" method="POST">
```

```
  <label>이름 : <input type="text" name="name" placeholder="이름을 입력"></label><br>
```

```
  <label>나이 : <input type="text" name="age" placeholder="나이를 입력"></label><br>
```

```
  <label>전화번호 : <input type="text" name="phone" placeholder="전화번호 입력"></label><br>
```

```
  <label>성향 : <input type="radio" name="active" value="1" checked> 활동적
```

```
             <input type="radio" name="active" value="0"> 정적
```

```
</label><br>
```

```
  <input type="submit" value="전송">
```

```
</form>
```

```
</div>
```

```
</body>
```

```
</html>
```



데이터 전송하기

(20) 하이퍼 링크로 하나의 데이터 전송

[데이터 전송1](#) [데이터 전송2](#) [데이터 전송3](#)

(21) Form에 입력된 데이터 전송

이름 :

나이 :

전화번호 :

성향 : ☒ 활동적 ☐ 정적

웹화면

Log 출력 결과

```
전송된 데이터 = Friend(name=Gildong, age=0, phone=null, active=false)
전송된 데이터 = Friend(name=Gildong, age=33, phone=null, active=false)
전송된 데이터 = Friend(name=null, age=0, phone=null, active=false)
전송된 데이터 = Friend(name=김길동, age=27, phone=010-2223-0101, active=true)
```

□ Fragment (1)

• Fragment란?

- 홈페이지를 만들 때 여러 페이지 내에서 중복되는 화면을 별도의 파일로 분리한 후 클라이언트에 보여질 때에는 하나의 화면으로 합쳐져서 보여지도록 하는 기능이다.
- 상단에 반복되는 gnb나 반복적인 footer를 만들 때 주로 사용된다.

• 만드는 방법

- 반복되는 문서코드를 별도의 파일(html)로 저장
- 반복되는 영역에 th:fragment 속성을 이용하여 이름 지정한다.
- 합쳐지는 파일에서 전달되는 데이터가 있을 경우 함수의 전달인자와 같이 변수를 이용해 받을 수 있다.

```
<ul class="navigation" th:fragment="top-menu(uid)">
  <li><a class="navi" th:text="|${uid} 홈|" th:href="@{/}">홈</a></li>
  <li><a class="navi" href="#">방명록</a></li>
  <li><a class="navi" href="#">로그인</a></li>
  <li><a class="navi" href="#">회원가입</a></li>
  <li><a class="navi" href="#">게시판</a></li>
</ul>
```

□ Fragment (2)

- 호출하는 방법

- 반복되는 코드를 삽입하는 문서에서는 th:replace()를 이용해 반복문서를 삽입한다.

```
<div th:replace="fragment/navigator :: top-menu(uid=사오정)"></div>
```

- 삽입하는 방법

```
<div th:replace="폴더명/파일명 :: 플래그먼트명"></div>
```

```
<div th:replace="폴더명/파일명 :: 플래그먼트명(변수명=값)"></div>
```

[실습] Fragment의 사용 (1) - Controller / HTML(Fragment 분리)

```
@Controller
@RequestMapping("/display")
public class FragmentController {

    @GetMapping("/fragment")
    public String fragment() {
        return "display/fragmentTest";
    }
}
```

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>navigator</title>
</head>
<body>
<div>
    <ul class="navigation" th:fragment="top-menu(uid)">
        <li><a class="navi" th:text="|${uid} 홈|" th:href="@{}/">홈</a></li>
        <li><a class="navi" href="#">방명록</a></li>
        <li><a class="navi" href="#">로그인</a></li>
        <li><a class="navi" href="#">회원가입</a></li>
        <li><a class="navi" href="#">게시판</a></li>
    </ul>
</div>
</body>
</html>
```

[실습] Fragment의 사용 (2) -HTML(Fragment 호출)

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
<link rel="stylesheet" th:href="@{/style/top.css}">
</head>
<body>
<div th:replace="fragment/navigator :: top-menu(uid=사오정)"></div>
<div class="container">
  <h2 class="main">[게시글 목록]</h2>
  <!-- 게시글 검색 -->
  <form th:action="@{/}" method="GET">
    <select name="searchItem">
      <option value="title">제목</option>
      <option value="usrId">글쓴이</option>
      <option value="content">글내용</option>
    </select>
    <input type="text" name="searchWord" >
    <input class="btn" type="submit" value="검색">
  </form>
</div>
</body>
</html>
```

사오정 홈 방명록 로그인 회원가입 게시판

[게시글 목록]

제목 ▼

The End!