

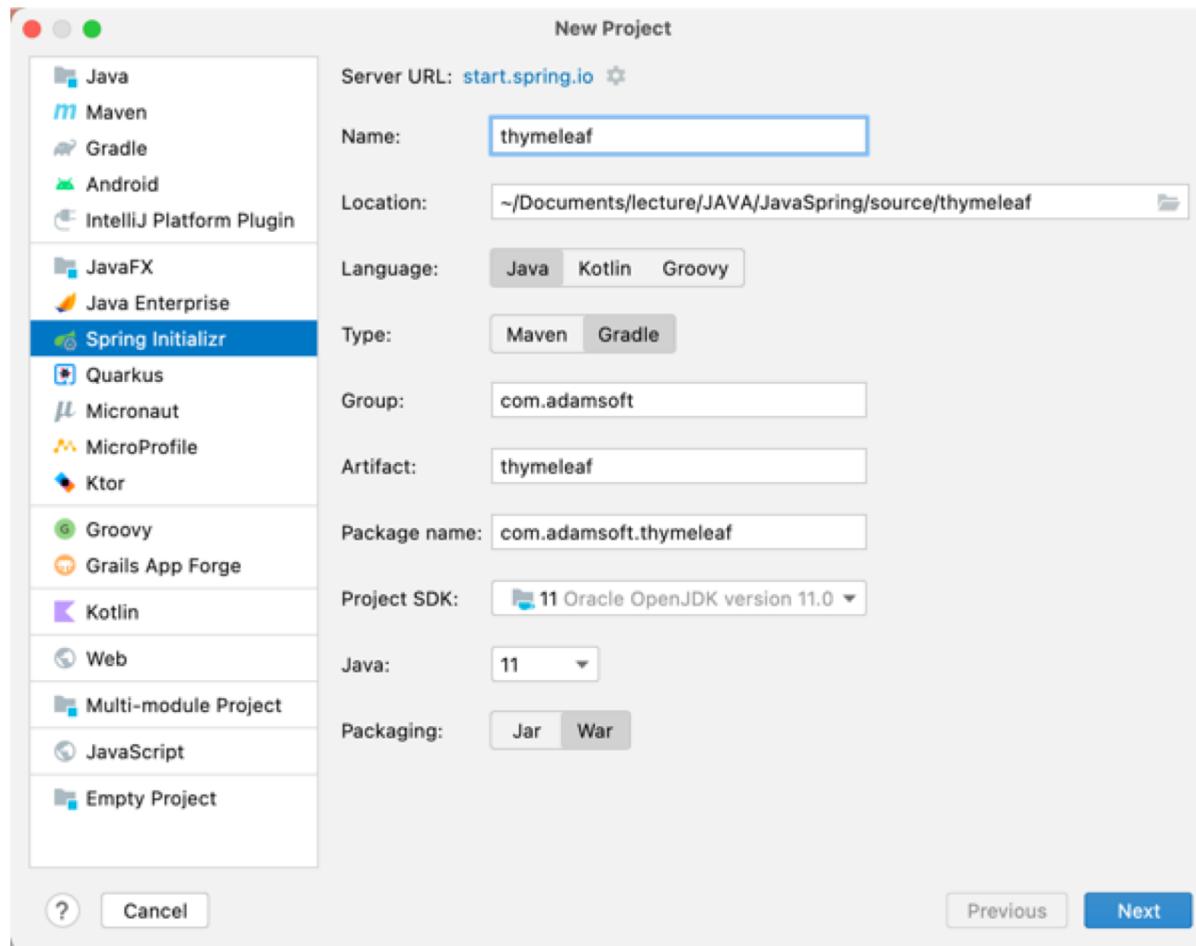
Thymeleaf

Spring Boot View

- ❖ Spring Boot 에서 화면 출력
 - ✓ JSP 활용 – 설정을 해야 함
 - ✓ 템플릿 엔진 사용
 - Thymeleaf
 - Velocity
 - FreeMarker
 - Mustache
 - Groovy
 - ✓ REST API 서버를 구성하고 자바스크립트의 ajax를 이용하거나 이를 편리하게 사용할 수 있도록 해주는 라이브러리(jquery, react, vue, angular 등)를 이용

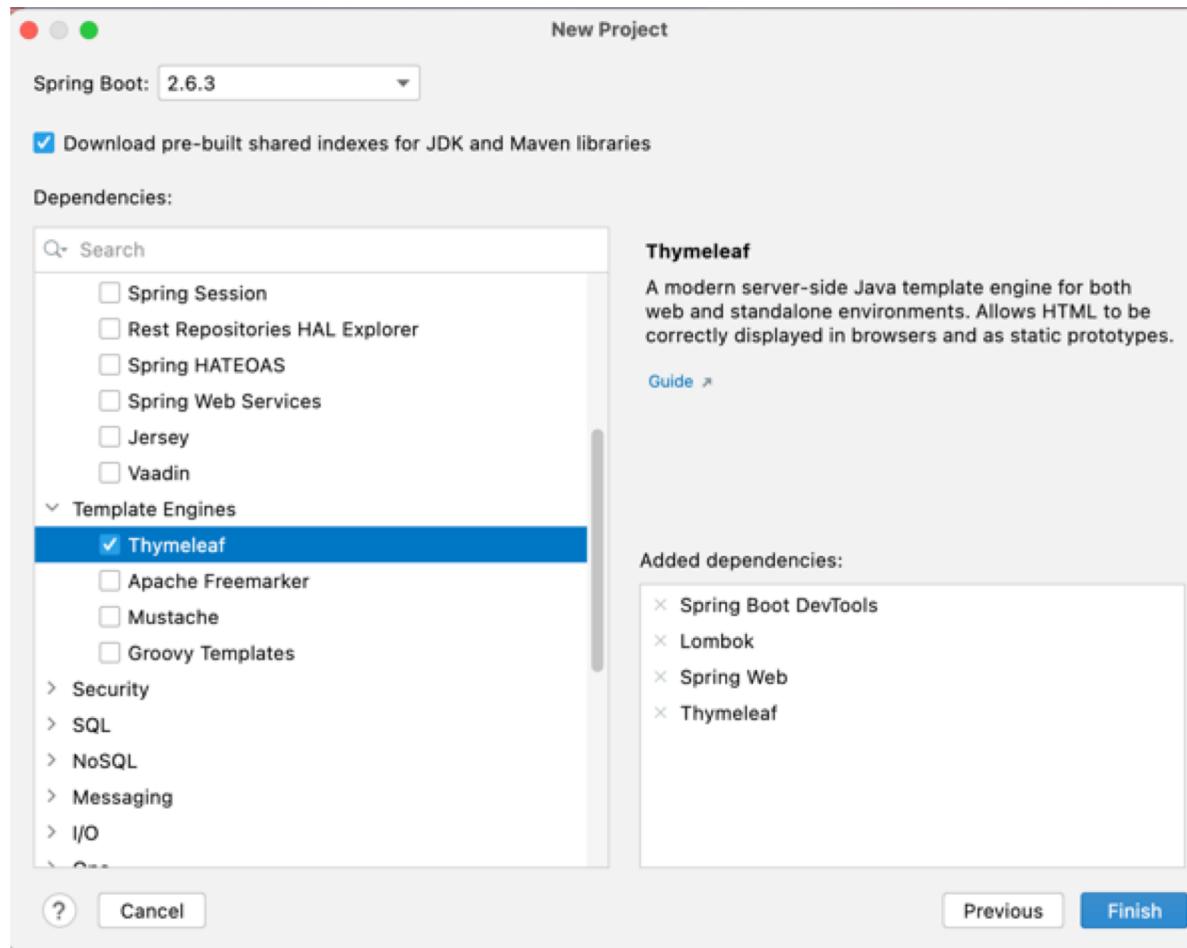
Spring Boot View

- ❖ Spring Boot 프로젝트 생성



Spring Boot View

❖ Spring Boot 프로젝트 생성



JSP

- ❖ Spring Boot 에서 JSP
 - ✓ JSP 사용을 위한 의존성 설정

- pom.xml

```
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
</dependency>
<dependency>
    <groupId>org.apache.tomcat.embed</groupId>
    <artifactId>tomcat-embed-jasper</artifactId>
</dependency>
```

- build.gradle

```
implementation 'javax.servlet:jstl'
implementation 'org.apache.tomcat.embed:tomcat-embed-jasper'
```

JSP

- ❖ Spring Boot 에서 JSP
 - ✓ application.properties 파일에 jsp 디렉토리 위치 설정

```
spring.mvc.view.prefix=/WEB-INF/views/  
spring.mvc.view.suffix=.jsp
```
 - ```
spring.thymeleaf.prefix=classpath:/templates/
spring.thymeleaf.suffix=.html
spring.thymeleaf.cache=false
```
  - ```
spring.thymeleaf.view-names=thymeleaf/*
```

JSP

- ❖ Spring Boot 에서 JSP

- ✓ controller 패키지를 생성하고 PageController 클래스를 생성하고 요청 처리 메서드 작성

```
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.web.bind.annotation.GetMapping;  
  
import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.List;  
import java.util.Map;
```

JSP

- ❖ Spring Boot에서 JSP

- ✓ controller 패키지를 생성하고 PageController 클래스를 생성하고 요청 처리 메서드 작성

```
@Controller  
public class PageController {  
    @GetMapping("/")  
    public String main(Model model){  
        Map<String, Object> map = new HashMap<>();  
        map.put("Language", "Java");  
        map.put("IDE", "IntelliJ");  
        map.put("BuildTool", "Gradle");  
        model.addAttribute("map", map);  
        List<String> list = new ArrayList<>();  
        list.add("BackEnd Developer");  
        list.add("FrontEnd Developer");  
        list.add("DBA");  
        list.add("Operator");  
        list.add("BigData");  
        list.add("DevOps");  
        list.add("AI");  
        list.add("MLOps");  
        model.addAttribute("list", list);  
        return "main";  
    }  
}
```

JSP

- ❖ Spring Boot 에서 JSP

- ✓ src/main 디렉토리 안에 webapp 디렉토리를 만들고 그 안에 WEB-INF 디렉토리를 추가하고 다시 views 디렉토리를 생성하고 main.jsp 파일을 생성한 후 작성

```
<%@ page language="java" contentType="text/html; charset=UTF-8"  
    pageEncoding="UTF-8"%>  
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>  
  
<!DOCTYPE html>  
<html lang="ko">  
<head>  
    <title>Spring Boot 에서 JSP</title>  
    <meta charset="UTF-8" />  
    <style>  
        table, tr, td, th {  
            border: 1px solid #444444;  
        }  
    </style>  
</head>
```

JSP

❖ Spring Boot에서 JSP

- ✓ src/main 디렉토리 안에 webapp 디렉토리를 만들고 그 안에 WEB-INF 디렉토리를 추가하고 다시 views 디렉토리를 생성하고 main.jsp 파일을 생성한 후 작성

```
<body>
<div>
    <table>
        <tr>
            <th>언어</th>
            <th>통합 개발 환경</th>
            <th>빌드 도구</th>
        </tr>
        <tr>
            <td>${map.Language}</td>
            <td>${map.IDE}</td>
            <td>${map.BuildTool}</td>
        </tr>
    </table>
</div>
```

JSP

- ❖ Spring Boot에서 JSP

- ✓ src/main 디렉토리 안에 webapp 디렉토리를 만들고 그 안에 WEB-INF 디렉토리를 추가하고 다시 views 디렉토리를 생성하고 main.jsp 파일을 생성한 후 작성

```
<div>
    <table>
        <c:forEach items="${list}" var="task">
            <tr>
                <td>${task}</td>
            </tr>
        </c:forEach>
    </table>
</div>

</body>
</html>
```

Thymeleaf

❖ Thymeleaf

- ✓ 화면 출력을 위한 템플릿 엔진 중의 하나
- ✓ 장점
 - 데이터를 JSP와 유사하게 \${}를 이용해서 출력
 - Model에 담긴 객체를 화면에서 JavaScript로 처리하기 편리
 - 연산이나 포맷과 관련된 기능을 추가적인 개발없이 지원
 - natural templates - html 파일로 출력 가능한데 서버 사이드 랜더링을 하지 않고 브라우저에 띄워도 정상적인 화면을 출력을 할 수 있음
- ✓ <https://www.thymeleaf.org/>
- ✓ Request, HttpSession, Application 에 저장된 데이터를 태그에 출력하고자 할 때는 태그 안에 th:text 속성을 추가한 후 데이터를 EL 형태로 설정하면 됨

Thymeleaf

❖ Spring Boot Devtools 의 주요 기능

- ✓ Automatic Restart: classpath에 있는 파일이 변경될 때마다 애플리케이션을 자동으로 재시작해서 개발자가 소스 수정 후 애플리케이션을 재실행하는 과정을 줄일 수 있으므로 생산성을 향상시킬 수 있음
- ✓ Live Reload: 정적 자원(html, css, js) 수정 시 새로 고침 없이 바로 적용
- ✓ Property Defaults: Thymeleaf는 기본적으로 성능을 향상시키기 위해서 캐싱 기능을 사용하는데 개발하는 과정에서 캐싱 기능을 사용한다면 수정한 소스가 제대로 반영되지 않을 수 있기 때문에 cache의 기본값을 false로 설정할 수 있음

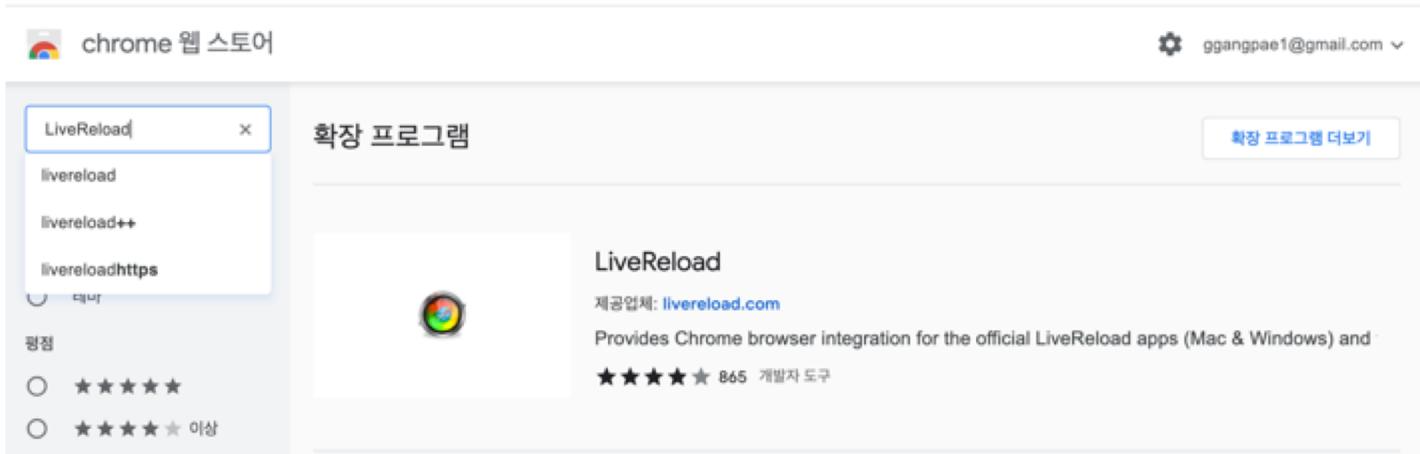
Thymeleaf

- ❖ Spring Boot Devtools 의 주요 기능
 - ✓ Live Reload 적용
 - application.properties 파일에 설정

```
#Live Reload 기능 활성화  
spring.devtools.livereload.enabled=true
```

Thymeleaf

- ❖ Spring Boot Devtools 의 주요 기능
 - ✓ Live Reload 적용
 - 크롬 웹 스토어에서 LiveReload를 검색해서 설치



Thymeleaf

❖ Thymeleaf 프로젝트

- ✓ 프로젝트에 Thymeleaf 라는 템플릿의 의존성을 설정해야 함
- ✓ 변경 후에 만들어진 결과를 캐싱하지 않도록 설정하는데 개발 환경에서는 캐싱 기능을 꺼두는 것을 권장하고 운영 환경에서는 사용하는 것이 좋음 – application.properties 파일의 이전 내용을 모두 삭제하고 아래 코드를 추가

```
spring.thymeleaf.cache=false
```

- ✓ PageController 클래스에 작성

```
@GetMapping("/ex1")
public void ex1(){
    System.out.println("ex1");
}
```

Thymeleaf

- ❖ Thymeleaf 프로젝트

- ✓ src/main/resources/templates 디렉토리에 ex1.html 파일을 만들고 작성 – 기존의 속성 앞에 th:를 붙이고 사용

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>ex1</title>
</head>
<body>
    <h1 th:text="${'Hello Thymeleaf'}"></h1>
</body>
</html>
```

Thymeleaf

- ❖ Thymeleaf 프로젝트

- ✓ application.properties 파일 수정

```
#spring.mvc.view.prefix=/WEB-INF/views/
```

```
#spring.mvc.view.suffix=.jsp
```

```
#spring.thymeleaf.prefix=classpath:/templates/
```

```
#spring.thymeleaf.suffix=.html
```

```
#spring.thymeleaf.cache=false
```

```
#spring.thymeleaf.view-names=thymeleaf/*
```

```
spring.thymeleaf.cache=false
```

```
spring.devtools.livereload.enabled=true
```

Thymeleaf

- ❖ Thymeleaf 프로젝트
 - ✓ 실행 후 브라우저에 localhost:8080/ex1 에 입력하고 확인
-

Hello Thymeleaf

Thymeleaf

- ❖ Thymeleaf 데이터 출력
 - ✓ 속성이 아닌 곳에서의 데이터 출력
[[\${데이터}]]
- ✓ 반복문
 - th:each = "변수: \${목록} "
 - 반복문을 사용하면 state 객체가 같이 생성되는데 이를 이용하면 순번이나 인덱스 번호, 카운트, 훌수/짝수 등을 지정할 수 있음
- ✓ 분기문
 - ✓ th:if ~ unless 를 이용하면 조건문을 사용할 수 있는데 th:if 와 th:unless는 별도로 작성 가능
 - ✓ th:switch 와 th:case 사용 가능
 - ✓ 삼항 연산자 사용이 가능한데 마지막 항은 생략이 가능

Thymeleaf

- ❖ Thymeleaf 데이터 출력
 - ✓ template 디렉토리에 main.html 파일을 생성하고 작성

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <style>
        table, tr, td, th {
            border: 1px solid #444444;
        }
    </style>
</head>
```

Thymeleaf

- ❖ Thymeleaf 데이터 출력
 - ✓ template 디렉토리에 main.html 파일을 생성하고 작성

```
<body>
<div>
    <table>
        <tr>
            <th>언어</th>
            <th>통합 개발 환경</th>
            <th>빌드 도구</th>
        </tr>
        <tr>
            <td>[ ${map.Language}]</td>
            <td>[ ${map.IDE}]</td>
            <td>[ ${map.BuildTool}]</td>
        </tr>
    </table>
</div>
```

Thymeleaf

- ❖ Thymeleaf 데이터 출력
 - ✓ template 디렉토리에 main.html 파일을 생성하고 작성

```
<div>
    <table>
        <tr th:each="task : ${list}">
            <td>[${task}]</td>
        </tr>
    </table>
</div>

</body>
</html>
```

Thymeleaf

- ❖ Thymeleaf 데이터 출력

- ✓ 기본패키지에 domain 패키지를 생성한 후 SampleVO 클래스를 생성하고 작성

```
import java.time.LocalDateTime;
```

```
import lombok.Builder;
```

```
import lombok.Data;
```

```
@Data
```

```
@Builder(toBuilder = true)
```

```
public class SampleVO {
```

```
    private Long sno;
```

```
    private String first;
```

```
    private String last;
```

```
    private LocalDateTime regTime;
```

```
}
```

Thymeleaf

- ❖ Thymeleaf 데이터 출력
 - ✓ PageController에 요청 처리 메서드 작성

```
@GetMapping("/ex2")
public void ex2(Model model){
    List<SampleVO> list = IntStream.rangeClosed(1,20).asLongStream().mapToObj(i -> {
        SampleVO vo = SampleVO.builder()
            .sno(i)
            .first("First.." + i)
            .last("Last.." + i)
            .regTime(LocalDateTime.now())
            .build();
        return vo;
    }).collect(Collectors.toList());

    model.addAttribute("list", list);
}
```

Thymeleaf

- ❖ Thymeleaf 데이터 출력
 - ✓ template 디렉토리에 ex2.html 파일을 생성하고 작성

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>List</title>
</head>
<body>
    <ul>
        <li th:each="vo : ${list}">
            [[${vo}]]
        </li>
    </ul>
</body>
</html>
```

Thymeleaf

- ❖ Thymeleaf 데이터 출력

- ✓ 실행 후 브라우저에 localhost:8080/ex2 에 입력하고 확인

- SampleVO(sno=0, first=First..0, last=Last..0, regTime=2022-01-10T07:37:56.399203)
 - SampleVO(sno=1, first=First..1, last=Last..1, regTime=2022-01-10T07:37:56.399301)
 - SampleVO(sno=2, first=First..2, last=Last..2, regTime=2022-01-10T07:37:56.399317)
 - SampleVO(sno=3, first=First..3, last=Last..3, regTime=2022-01-10T07:37:56.399327)
 - SampleVO(sno=4, first=First..4, last=Last..4, regTime=2022-01-10T07:37:56.399335)
 - SampleVO(sno=5, first=First..5, last=Last..5, regTime=2022-01-10T07:37:56.399358)
 - SampleVO(sno=6, first=First..6, last=Last..6, regTime=2022-01-10T07:37:56.399364)
 - SampleVO(sno=7, first=First..7, last=Last..7, regTime=2022-01-10T07:37:56.399371)
 - SampleVO(sno=8, first=First..8, last=Last..8, regTime=2022-01-10T07:37:56.399377)
 - SampleVO(sno=9, first=First..9, last=Last..9, regTime=2022-01-10T07:37:56.399383)

Thymeleaf

- ❖ Thymeleaf 데이터 출력
 - ✓ template 디렉토리에 ex2.html 파일을 수정

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>List</title>
</head>
<body>
    <ul>
        <li th:each="vo, state : ${list}">
            [[${state.index}]] --- [[${vo}]]
        </li>
    </ul>
</body>
</html>
```

Thymeleaf

- ❖ Thymeleaf 데이터 출력
 - ✓ 실행 후 브라우저에 localhost:8080/ex2 에 입력하고 확인

- 0 --- SampleVO(sno=0, first=First..0, last=Last..0, regTime=2022-01-10T07:51:08.841608)
- 1 --- SampleVO(sno=1, first=First..1, last=Last..1, regTime=2022-01-10T07:51:08.841656)
- 2 --- SampleVO(sno=2, first=First..2, last=Last..2, regTime=2022-01-10T07:51:08.841667)
- 3 --- SampleVO(sno=3, first=First..3, last=Last..3, regTime=2022-01-10T07:51:08.841677)
- 4 --- SampleVO(sno=4, first=First..4, last=Last..4, regTime=2022-01-10T07:51:08.841686)
- 5 --- SampleVO(sno=5, first=First..5, last=Last..5, regTime=2022-01-10T07:51:08.841694)
- 6 --- SampleVO(sno=6, first=First..6, last=Last..6, regTime=2022-01-10T07:51:08.841702)
- 7 --- SampleVO(sno=7, first=First..7, last=Last..7, regTime=2022-01-10T07:51:08.841710)
- 8 --- SampleVO(sno=8, first=First..8, last=Last..8, regTime=2022-01-10T07:51:08.841717)
- 9 --- SampleVO(sno=9, first=First..9, last=Last..9, regTime=2022-01-10T07:51:08.841725)

Thymeleaf

- ❖ Thymeleaf 데이터 출력
 - ✓ template 디렉토리에 ex2.html 파일을 수정

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>List</title>
</head>
<body>
    <ul>
        <li th:each="vo, state : ${list}" th:if="${vo.sno % 5 == 0}">
            [[${state.index}]] --- [[${vo}]]
        </li>
    </ul>
</body>
</html>
```

Thymeleaf

- ❖ Thymeleaf 데이터 출력
 - ✓ 실행 후 브라우저에 localhost:8080/ex2 에 입력하고 확인

- 0 --- SampleVO(sno=0, first=First..0, last=Last..0, regTime=2022-01-10T07:55:50.385959)
- 5 --- SampleVO(sno=5, first=First..5, last=Last..5, regTime=2022-01-10T07:55:50.386040)

Thymeleaf

- ❖ Thymeleaf 데이터 출력
 - ✓ template 디렉토리에 ex2.html 파일을 수정

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>List</title>
</head>
<body>
    <ul>
        <li th:each="vo, state : ${list}">
            <span th:if="${vo.sno % 5 == 0}" th:text="${'-----' + vo.sno}"></span>
            <span th:unless="${vo.sno % 5 == 0}" th:text="${vo.first}"></span>
        </li>
    </ul>
</body>
</html>
```

Thymeleaf

- ❖ Thymeleaf 데이터 출력
 - ✓ 실행 후 브라우저에 localhost:8080/ex2 에 입력하고 확인
 - -----0
 - First..1
 - First..2
 - First..3
 - First..4
 - -----5
 - First..6
 - First..7
 - First..8
 - First..9

Thymeleaf

- ❖ Thymeleaf 데이터 출력
 - ✓ template 디렉토리에 ex2.html 파일을 수정

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>List</title>
    <style>
        .target{
            background-color: red;
        }
    </style>
</head>
```

Thymeleaf

- ❖ Thymeleaf 데이터 출력
 - ✓ template 디렉토리에 ex2.html 파일을 수정

```
<body>
<ul>
    <li th:each="vo, state : ${list}" th:text="${vo.sno % 5 == 0}?${vo.sno}: ${vo.first}">
        </li>
    </ul>
    <ul>
        <li th:each="vo, state : ${list}" th:class="${vo.sno % 5 == 0}?'target'" th:text="${vo}">
            </li>
        </ul>
    </body>
</html>
```

Thymeleaf

- ❖ Thymeleaf 데이터 출력
 - ✓ 실행 후 브라우저에 localhost:8080/ex2 에 입력하고 확인
 - 0
 - First..1
 - First..2
 - First..3
 - First..4
 - 5
 - First..6
 - First..7
 - First..8
 - First..9
 - SampleVO(sno=0, first=First..0, last=Last..0, regTime=2022-01-10T08:49:55.259293)
 - SampleVO(sno=1, first=First..1, last=Last..1, regTime=2022-01-10T08:49:55.259343)
 - SampleVO(sno=2, first=First..2, last=Last..2, regTime=2022-01-10T08:49:55.259352)
 - SampleVO(sno=3, first=First..3, last=Last..3, regTime=2022-01-10T08:49:55.259361)
 - SampleVO(sno=4, first=First..4, last=Last..4, regTime=2022-01-10T08:49:55.259368)
 - SampleVO(sno=5, first=First..5, last=Last..5, regTime=2022-01-10T08:49:55.259375)
 - SampleVO(sno=6, first=First..6, last=Last..6, regTime=2022-01-10T08:49:55.259382)
 - SampleVO(sno=7, first=First..7, last=Last..7, regTime=2022-01-10T08:49:55.259388)
 - SampleVO(sno=8, first=First..8, last=Last..8, regTime=2022-01-10T08:49:55.259395)
 - SampleVO(sno=9, first=First..9, last=Last..9, regTime=2022-01-10T08:49:55.259401)

Thymeleaf

- ❖ Thymeleaf 데이터 출력
 - ✓ th:block
 - th:block은 별도의 태그가 필요하지 않기 때문에 반드시 태그에 붙어서 th:text나 th:value 등을 써야 하는 제약이 없음

Thymeleaf

- ❖ Thymeleaf 데이터 출력

- ✓ th:block

- ex2.html 파일 수정

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>List</title>
    <style>
        .target{
            background-color: red;
        }
    </style>
</head>
<body>
    <ul>
        <th:block th:each="vo:${list}">
            <li th:text="${vo.sno % 5 == 0}?${vo.sno}: ${vo.first}"></li>
        </th:block>
    </ul>
</body>
</html>
```

Thymeleaf

- ❖ Thymeleaf 데이터 출력

- ✓ th:block

- ex2.html 파일 수정

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>List</title>
</head>
<body>
<ul>
    <li th:each="vo, state : ${list}">
        <th:block th:switch="${vo.sno % 5 == 0}">
            <span th:case=true th:text="${'-----' + vo.sno}"></span>
            <span th:case=false th:text="${vo.first}"></span>
        </th:block>
    </li>
</ul>
</body>
</html>
```

Thymeleaf

- ❖ Thymeleaf 데이터 출력
 - ✓ inline
 - 자바스크립트에서 넘어온 데이터를 사용하면 문자열은 자동으로 문자열로 처리됨
 - List 나 DTO 객체는 JSON 포맷의 문자열로 변환되어서 파싱 된 결과가 전달됨

Thymeleaf

- ❖ Thymeleaf 데이터 출력

- ✓ inline

- PageController 클래스에 새로운 요청을 처리하는 메서드를 생성

```
@GetMapping("/inline")
public String exInline(RedirectAttributes redirectAttributes){
    SampleVO vo =
        SampleVO.builder().sno(100L).first("First..100").last("Last..100").regTime(LocalDateTime.now()).build();
    redirectAttributes.addFlashAttribute("result", "success");
    redirectAttributes.addFlashAttribute("vo", vo);
    return "redirect:/ex3";
}
```

```
@GetMapping("/ex3")
public void ex3(){}
```

Thymeleaf

- ❖ Thymeleaf 데이터 출력
 - ✓ inline
 - template 디렉토리에 ex3.html 파일을 생성하고 작성

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset='UTF-8'>
    <title>Title</title> </head>
<body>
    <h1 th:text="${result}"></h1>
    <h2 id="result"></h2>
    <h1 th:text="${vo}"></h1>
    <h2 id="vo"></h2>

    <script th:inline="javascript">
        let msg = [[${result}]];
        let vo = [[${vo}]];
        document.getElementById("result").innerHTML = msg;
        document.getElementById("vo").innerHTML = vo.sno;
    </script>
</body>
</html>
```

Thymeleaf

- ❖ Thymeleaf 데이터 출력
 - ✓ inline
 - 실행 후 브라우저에 localhost:8080/inline 에 입력하고 확인

success

success

SampleVO(sno=100, first=First..100, last=Last..100, regTime=2022-01-10T15:38:54.052994)

100

Thymeleaf

- ❖ Thymeleaf 데이터 출력
 - ✓ 링크 처리
 - href 속성에 @{ }를 이용해서 설정
 - 파라미터를 전달할 때는 (파라미터이름=\${출력할 데이터})
 - path 로 전달하고자 하는 경우에는 /뒤에 {임시변수}(임시변수 = \${데이터} 형태로 작성

Thymeleaf

- ❖ Thymeleaf 데이터 출력
 - ✓ 링크 처리
 - PageController 의 ex02를 처리하는 부분을 수정

```
@GetMapping({"/ex2", "/exlink"})
public void ex2(Model model){
    List<SampleVO> list = new ArrayList<>();
    for(long i=0; i<10; i=i+1) {
        SampleVO vo = SampleVO.builder()
            .sno(i)
            .first("First.." + i)
            .last("Last.." + i)
            .regTime(LocalDateTime.now())
            .build();
        list.add(vo);
    }
    model.addAttribute("list", list);
    System.out.println(list);
}
```

Thymeleaf

- ❖ Thymeleaf 데이터 출력
 - ✓ 링크 처리
 - template 디렉토리에 exlink.html 파일을 생성하고 작성

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Link</title>

</head>
<body>
<ul>
    <li th:each="vo:${list}">
        <a th:href="@{/exview}">[${vo.first}]</a>
        <a th:href="@{/exview(sno=${vo.sno})}">[${vo.first}]</a>
        <a th:href="@{/exview/{sno}(sno = ${vo.sno})}">[${vo.first}]</a>
    </li>
</ul>

</body>
</html>
```

Thymeleaf

- ❖ Thymeleaf 데이터 출력
 - ✓ 링크 처리
 - 실행 후 브라우저에 localhost:8080/exlink 에 입력하고 확인

- [\\${vo.first} First..0 First..0](#)
- [\\${vo.first} First..1 First..1](#)
- [\\${vo.first} First..2 First..2](#)
- [\\${vo.first} First..3 First..3](#)
- [\\${vo.first} First..4 First..4](#)
- [\\${vo.first} First..5 First..5](#)
- [\\${vo.first} First..6 First..6](#)
- [\\${vo.first} First..7 First..7](#)
- [\\${vo.first} First..8 First..8](#)
- [\\${vo.first} First..9 First..9](#)

Thymeleaf

- ❖ Thymeleaf 데이터 포맷
 - ✓ 데이터의 포맷 설정
 - 숫자의 경우는 #numbers를 이용해서 숫자의 포맷을 설정
 - 날짜의 경우는 #temporals를 이용해서 날짜의 포맷을 설정하는데 build.gradle 파일에 아래 의존성이 추가되어야 함
implementation group: 'org.thymeleaf.extras', name: 'thymeleaf-extras-java8time'

Thymeleaf

- ❖ Thymeleaf 데이터 포맷
 - ✓ 데이터의 포맷 설정
 - build.gradle 파일의 dependencies 에 아래 내용을 추가하고 적용
implementation group: 'org.thymeleaf.extras', name: 'thymeleaf-extras-java8time'

Thymeleaf

- ❖ Thymeleaf 데이터 포맷
 - ✓ 데이터의 포맷 설정
 - PageController 클래스에 요청 처리 메서드 수정
- ```
@GetMapping({"/ex2", "/exlink", "/exformat"})
public void ex2(Model model){
 List<SampleVO> list = new ArrayList<>();
 for(long i=0; i<10; i=i+1) {
 SampleVO vo = SampleVO.builder()
 .sno(i)
 .first("First.." + i)
 .last("Last.." + i)
 .regTime(LocalDateTime.now())
 .build();
 list.add(vo);
 }
 model.addAttribute("list", list);
 System.out.println(list);
}
```

# Thymeleaf

- ❖ Thymeleaf 데이터 포맷
  - ✓ 데이터의 포맷 설정
    - exformat.html 파일을 생성하고 작성

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
 <meta charset="UTF-8">
 <title>포맷</title>
</head>
<body>

 <li th:each="vo : ${list}">
 [[${#numbers.formatInteger(vo.sno, 5)}]] ---
 [[${#temporals.format(vo.regTime, 'yyyy/MM/dd')}]]

</body>
</html>
```

# Thymeleaf

- ❖ Thymeleaf 데이터 포맷
  - ✓ 데이터의 포맷 설정
    - 프로젝트를 실행하고 exformat 라고 입력하고 확인

- 00000 --- 2022/01/10
- 00001 --- 2022/01/10
- 00002 --- 2022/01/10
- 00003 --- 2022/01/10
- 00004 --- 2022/01/10
- 00005 --- 2022/01/10
- 00006 --- 2022/01/10
- 00007 --- 2022/01/10
- 00008 --- 2022/01/10
- 00009 --- 2022/01/10

# Thymeleaf

- ❖ Thymeleaf 레이아웃
  - ✓ 레이아웃 설정 방법
    - JSP의 include 와 같이 특정 부분을 외부 혹은 내부에서 가져와서 포함하는 형태
    - 특정한 부분을 파라미터로 전달해서 내용에 포함하는 형태
  - ✓ include 방식의 처리
    - 특정한 부분을 다른 내용으로 변경할 수 있는 th:insert나 th:replace 를 이용
    - th:replace를 이용하는 경우에는 기존의 내용을 완전히 대체하는 방식
    - th:insert의 경우에는 기존 내용의 바깥쪽 태그는 그대로 유지하면서 추가되는 방식

# Thymeleaf

- ❖ Thymeleaf 레이아웃
  - ✓ include 방식의 처리
    - PageController 클래스에 메서드 추가

```
@GetMapping("/exlayout1")
public void exlayout(){}
```

# Thymeleaf

- ❖ Thymeleaf 레이아웃
  - ✓ include 방식의 처리
    - template 디렉토리에 레이아웃에 사용될 파일들을 저장할 fragments 라는 디렉토리를 생성

# Thymeleaf

- ❖ Thymeleaf 레이아웃
  - ✓ include 방식의 처리
    - fragments 디렉토리에 fragment1.html 파일을 생성하고 작성

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
 <meta charset="UTF-8">
 <title>Title</title>
</head>
<body>
 <div th:fragment="part1">
 <h2>Part 1</h2>
 </div>
 <div th:fragment="part2">
 <h2>Part 2</h2>
 </div>
 <div th:fragment="part3">
 <h2>Part 3</h2>
 </div>
</body>
</html>
```

# Thymeleaf

- ❖ Thymeleaf 레이아웃

- ✓ include 방식의 처리

- ❑ templates 디렉토리에 exlayout1.html 파일을 생성하고 작성

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
 <meta charset="UTF-8">
 <title>Title</title>
</head>
<body>
 <h1>Fragment Test</h1>

 <h1>Layout 1 - 1</h1>
 <div th:replace="~/fragments/fragment1 :: part1" ></div>

 <h1>Layout 1 - 2</h1>
 <div th:replace="~/fragments/fragment1 :: part2" ></div>

 <h1>Layout 1 - 3</h1>
 <th:block th:replace="~/fragments/fragment1 :: part3" ></th:block>

</body>
</html>
```

# Thymeleaf

- ❖ Thymeleaf 레이아웃
  - ✓ include 방식의 처리
    - 애플리케이션을 실행하고 브라우저에 exlayout1 을 입력하고 확인

## Fragment Test

**Layout 1 - 1**

**Part 1**

**Layout 1 - 2**

**Part 2**

**Layout 1 - 3**

**Part 3**

# Thymeleaf

- ❖ Thymeleaf 레이아웃
  - ✓ include 방식의 처리
    - ❑ fragments 디렉토리에 fragment2.html 파일을 생성하고 작성

```
<div>
 <hr/>
 <h2>Fragment2 File</h2>
 <h2>Fragment2 File</h2>
 <h2>Fragment2 File</h2>
 <hr/>
</div>
```

# Thymeleaf

- ❖ Thymeleaf 레이아웃
  - ✓ include 방식의 처리
    - templates 디렉토리에 exlayout1.html 파일에 내용을 추가

```
<div style="border: 1px solid blue">
 <th:block th:replace="~/fragments/fragment2"></th:block>
</div>
```

# Thymeleaf

- ❖ Thymeleaf 레이아웃
  - ✓ include 방식의 처리
    - 애플리케이션을 실행하고 브라우저에 exlayout1 을 입력하고 확인

## Fragment Test

**Fragment2 File**

**Fragment2 File**

**Fragment2 File**

**Layout 1 - 1**

**Part 1**

**Layout 1 - 2**

**Part 2**

**Layout 1 - 3**

**Part 3**

# Thymeleaf

- ❖ Thymeleaf 레이아웃
  - ✓ 파라미터 방식의 처리
    - PageController 클래스의 메서드 수정

```
@GetMapping={"/exlayout1", "/exlayout2"}
public void exlayout(){}
```

# Thymeleaf

- ❖ Thymeleaf 레이아웃
  - ✓ 파라미터 방식의 처리
    - fragments 디렉토리에 fragment3.html 파일을 생성하고 작성

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
 <meta charset="UTF-8">
 <title>Title</title>
</head>
```

# Thymeleaf

- ❖ Thymeleaf 레이아웃
  - ✓ 파라미터 방식의 처리
    - fragments 디렉토리에 fragment3.html 파일을 생성하고 작성

```
<body>
<div th:fragment="target(first, second)">
 <style>
 .c1 {
 background-color: red;
 }
 .c2 {
 background-color: blue;
 }
 </style>

 <div class="c1">
 <th:block th:replace = "${first}"></th:block>
 </div>

 <div class="c2">
 <th:block th:replace = "${second}"></th:block>
 </div>
</body>
</html>
```

# Thymeleaf

- ❖ Thymeleaf 레이아웃
  - ✓ 파라미터 방식의 처리
    - templates 디렉토리에 exlayout2.html 파일을 생성하고 작성

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<th:block th:replace="~/fragments/fragment3:: target(~{this:: #ulFirst} ,
~{this::#ulSecond})">
 <ul id="ulFirst">
 JAVA
 JAVASCRIPT
 PYTHON

 <ul id="ulSecond">
 ORACLE
 MySQL
 MongoDB

</th:block>
```

# Thymeleaf

- ❖ Thymeleaf 레이아웃
  - ✓ 파라미터 방식의 처리
    - 애플리케이션을 실행하고 브라우저에 exlayout2 을 입력하고 확인

- JAVA
- JAVASCRIPT
- PYTHON

- ORACLE
- MySQL
- MongoDB

# Thymeleaf

- ❖ Thymeleaf 레이아웃
  - ✓ 레이아웃 템플릿 만들기
    - templates 디렉토리에 layout 디렉토리를 생성

# Thymeleaf

- ❖ Thymeleaf 레이아웃
  - ✓ 레이아웃 템플릿 만들기
    - layout 디렉토리에 layout1.html 파일을 생성하고 작성

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">

<th:block th:fragment="setContent(content)">

<head>
 <meta charset="UTF-8">
 <title>Title</title>
</head>

<body>

<style>
 * {
 margin: 0;
 padding: 0;
 }

```

# Thymeleaf

- ❖ Thymeleaf 레이아웃
  - ✓ 레이아웃 템플릿 만들기
    - layout 디렉토리에 layout1.html 파일을 생성하고 작성

```
.header {
 width:100vw;
 height: 20vh;
 background-color: aqua;
}
.content {
 width: 100vw;
 height: 70vh;
 background-color: lightgray;
}
.footer {
 width: 100vw;
 height: 10vh;
 background-color: green;
}
</style>
```

# Thymeleaf

- ❖ Thymeleaf 레이아웃
  - ✓ 레이아웃 템플릿 만들기
    - layout 디렉토리에 layout1.html 파일을 생성하고 작성

```
<div class="header">
 <h1>HEADER</h1>
</div>
<div class="content" >

 <th:block th:replace = "${content}">
 </th:block>

</div>

<div class="footer">
 <h1>FOOTER</h1>
</div>

</body>
</th:block>
</html>
```

# Thymeleaf

- ❖ Thymeleaf 레이아웃
  - ✓ 레이아웃 템플릿 만들기
    - PageController 클래스의 요청 처리 메서드 수정

```
@GetMapping={"/exlayout1", "/exlayout2", "/extemplate"}
public void exlayout(){}
```

# Thymeleaf

- ❖ Thymeleaf 레이아웃
  - ✓ 레이아웃 템플릿 만들기
    - templates 디렉토리에 extemplate.html 파일을 생성하고 작성

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">

<th:block th:replace="~/layout/layout1 :: setContent(~{this::content})">

<th:block th:fragment="content">

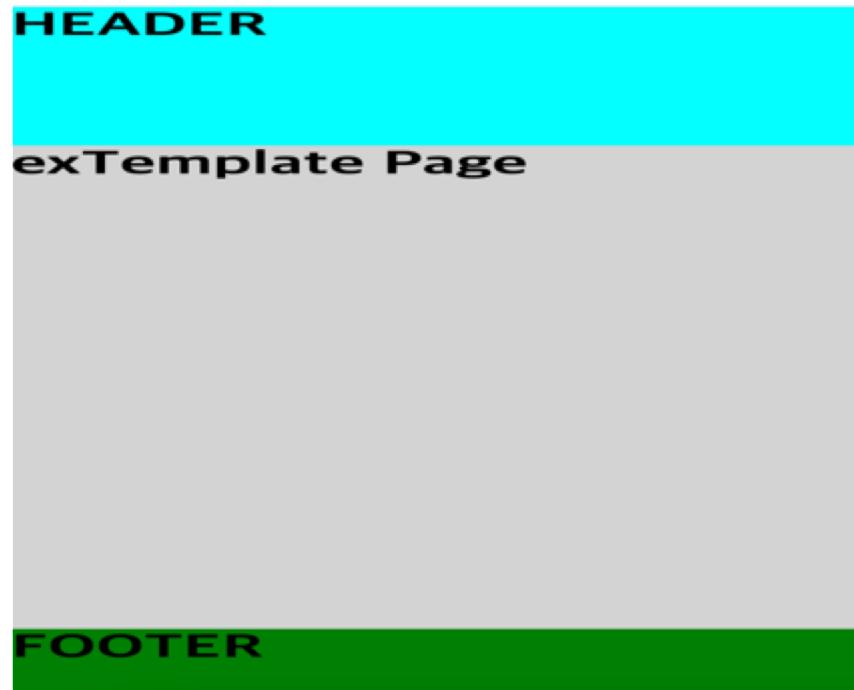
<h1>exTemplate Page</h1>

</th:block>

</th:block>
```

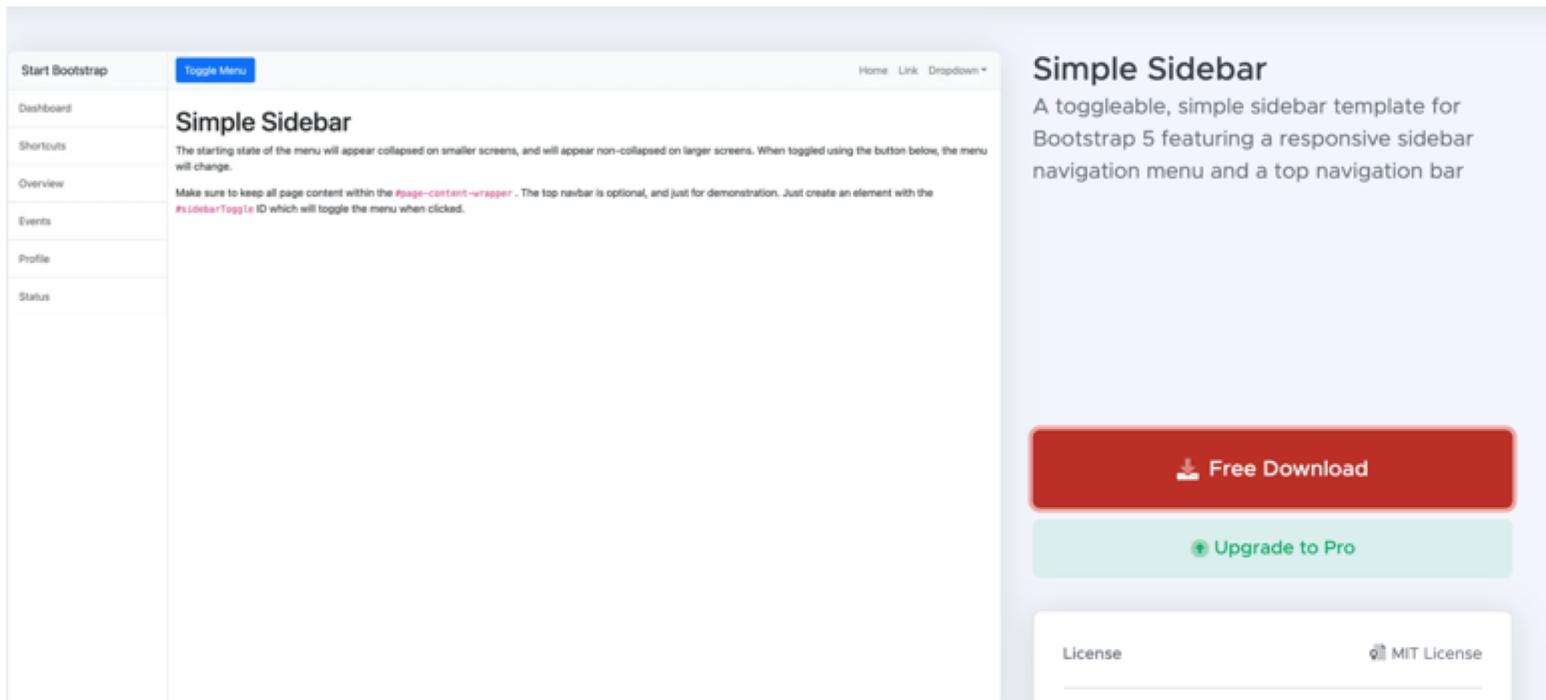
# Thymeleaf

- ❖ Thymeleaf 레이아웃
  - ✓ 레이아웃 템플릿 만들기
    - 애플리케이션을 실행하고 브라우저에 exTemplate 을 입력하고 확인



# Thymeleaf

- ❖ Thymeleaf 레이아웃
  - ✓ 부트스트랩 템플릿 적용하기
    - <https://startbootstrap.com/template/simple-sidebar>에서 sidebar 다운로드



# Thymeleaf

- ❖ Thymeleaf 레이아웃
  - ✓ 부트스트랩 템플릿 적용하기
    - 다운로드 받은 파일의 압축을 해제해서 압축을 해제한 파일 전체를 resources/static 디렉토리에 복사

# Thymeleaf

## ❖ Thymeleaf 레이아웃

### ✓ 부트스트랩 템플릿 적용하기

- ❑ template/layout 디렉토리에 basic.html 파일을 추가하고 복사한 파일의 index.html 파일의 내용을 전부 복사해서 붙여넣고 레이아웃 설정을 위해서 수정

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">

<th:block th:fragment="setContent(content)">

<head>
 <meta charset="utf-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no" />
 <meta name="description" content="" />
 <meta name="author" content="" />
 <title>Simple Sidebar - Start Bootstrap Template</title>
 <!-- Favicon-->
 <link rel="icon" type="image/x-icon" th:href="@{assets/favicon.ico}" />
 <!-- Core theme CSS (includes Bootstrap)-->
 <link th:href="@{/css/styles.css}" rel="stylesheet" />
```

# Thymeleaf

- ❖ Thymeleaf 레이아웃

- ✓ 부트스트랩 템플릿 적용하기

- template/layout 디렉토리에 basic.html 파일을 추가하고 복사한 파일의 index.html 파일의 내용을 전부 복사해서 붙여넣고 레이아웃 설정을 위해서 수정

```
<!-- Bootstrap core JS-->
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/js/bootstrap.bundle.min.js"
></script>
<!-- Core theme JS-->
<script th:src="@{/js/scripts.js}"></script>

<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>

</head>
```

# Thymeleaf

## ❖ Thymeleaf 레이아웃

### ✓ 부트스트랩 템플릿 적용하기

- ❑ template/layout 디렉토리에 basic.html 파일을 추가하고 복사한 파일의 index.html 파일의 내용을 전부 복사해서 붙여넣고 레이아웃 설정을 위해서 수정

```
<body>
<div class="d-flex" id="wrapper">
 <div class="border-end bg-white" id="sidebar-wrapper">
 <div class="sidebar-heading border-bottom bg-light">Start
 Bootstrap</div>
 <div class="list-group list-group-flush">
 Dashboard
 Shortcuts
 Overview
 Events
 Profile
 Status
 </div>
 </div>
</div>
```

# Thymeleaf

- ❖ Thymeleaf 레이아웃
  - ✓ 부트스트랩 템플릿 적용하기
    - template/layout 디렉토리에 basic.html 파일을 추가하고 복사한 파일의 index.html 파일의 내용을 전부 복사해서 붙여넣고 레이아웃 설정을 위해서 수정

```
<!-- Page content wrapper-->
<div id="page-content-wrapper">
 <!-- Top navigation-->
 <nav class="navbar navbar-expand-lg navbar-light bg-light border-bottom">
 <div class="container-fluid">
 <button class="btn btn-primary" id="sidebarToggle">Toggle
 Menu</button>
 <button class="navbar-toggler" type="button" data-bs-
 toggle="collapse" data-bs-target="#navbarSupportedContent" aria-
 controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle
 navigation"> </button>
```

# Thymeleaf

## ❖ Thymeleaf 레이아웃

### ✓ 부트스트랩 템플릿 적용하기

- template/layout 디렉토리에 basic.html 파일을 추가하고 복사한 파일의 index.html 파일의 내용을 전부 복사해서 붙여넣고 레이아웃 설정을 위해서 수정

```
<div class="collapse navbar-collapse" id="navbarSupportedContent">
 <ul class="navbar-nav ms-auto mt-2 mt-lg-0">
 <li class="nav-item active"><a class="nav-link"
 href="#">Home
 <li class="nav-item"><a class="nav-link"
 href="#">Link
 <li class="nav-item dropdown">
 <a class="nav-link dropdown-toggle" id="navbarDropdown"
 href="#" role="button" data-bs-toggle="dropdown" aria-haspopup="true" aria-
 expanded="false">Dropdown
 <div class="dropdown-menu dropdown-menu-end" aria-
 labelledby="navbarDropdown">
 Action
 Another
 action
 <div class="dropdown-divider"></div>
 Something else
 here

```

# Thymeleaf

- ❖ Thymeleaf 레이아웃

- ✓ 부트스트랩 템플릿 적용하기

- ❑ template/layout 디렉토리에 basic.html 파일을 추가하고 복사한 파일의 index.html 파일의 내용을 전부 복사해서 붙여넣고 레이아웃 설정을 위해서 수정

```
</div>

</div>
</div>
</nav>
<!-- Page content-->
<div class="container-fluid">
 <th:block th:replace = "${content}"></th:block>
</div>
</div>
</div>
</body>
</html>
```

# Thymeleaf

- ❖ Thymeleaf 레이아웃
  - ✓ 부트스트랩 템플릿 적용하기
    - PageController 클래스의 요청 처리 메서드 수정

```
@GetMapping={"/exlayout1", "/exlayout2", "/extemplate", "/exsidebar"}
public void exlayout(){}
```

# Thymeleaf

- ❖ Thymeleaf 레이아웃

- ✓ 부트스트랩 템플릿 적용하기

- ❑ templates 디렉토리에 exsidebar.html 파일을 생성하고 작성

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">

<th:block th:replace="~/layout/basic:: setContent(~{this::content})">

<th:block th:fragment="content">

<h1>exsidebar page</h1>

</th:block>

</th:block>
```

# Thymeleaf

- ❖ Thymeleaf 레이아웃
  - ✓ 부트스트랩 템플릿 적용하기
    - 애플리케이션을 실행하고 exsidebar를 입력하고 확인

