

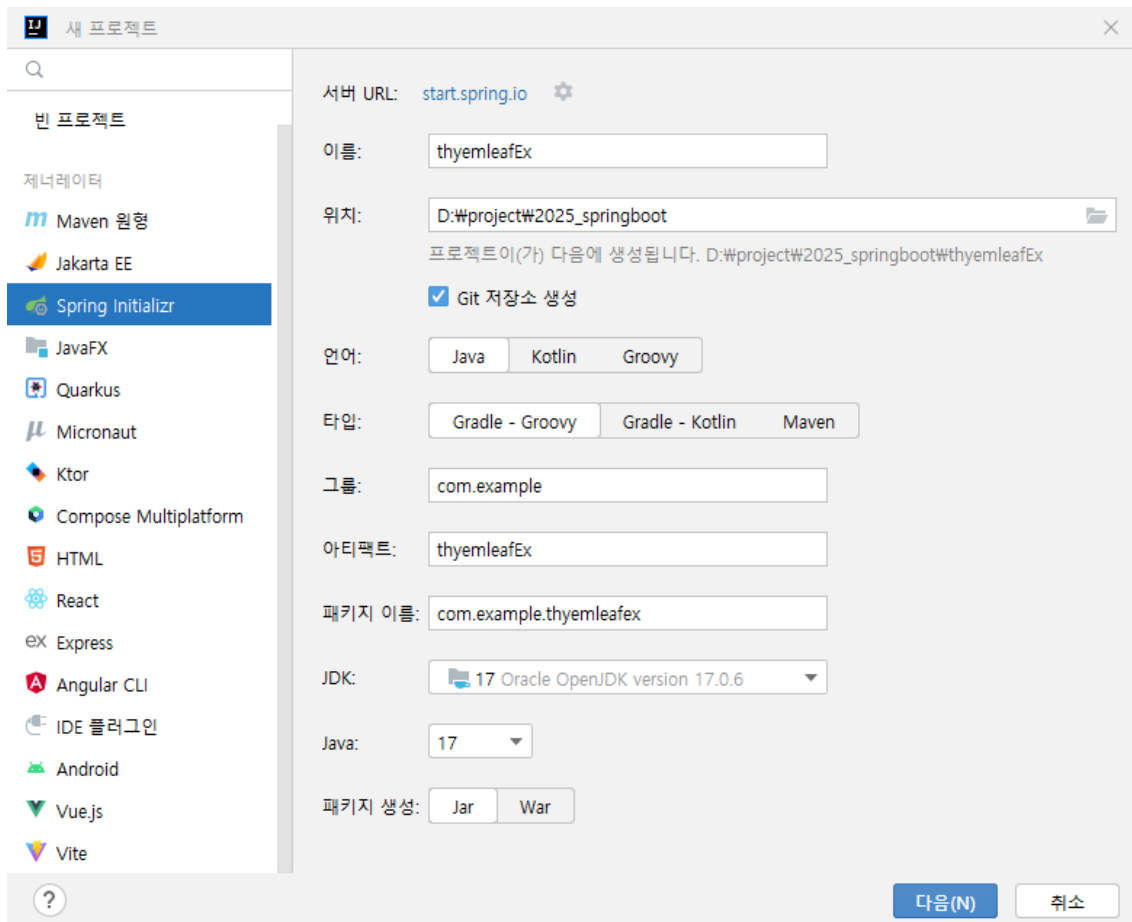
# 3. Thymeleaf

---

1. Thymeleaf Template Engines
2. Thymeleaf 기초
3. 반복문/제어문
4. Thymeleaf를 이용한 링크 처리
5. Thymeleaf의 JavaScript 인라인 처리
6. 레이아웃 기능
7. thymeleaf template 적용 실습

# 1. Thymeleaf Template Engines

## ❖ Thymeleaf 종속성 추가



새 프로젝트

빈 프로젝트

제너레이터

- Maven 원형
- Jakarta EE
- Spring Initializr**
- JavaFX
- Quarkus
- Micronaut
- Ktor
- Compose Multiplatform
- HTML
- React
- Express
- Angular CLI
- IDE 플러그인
- Android
- Vue.js
- Vite

서버 URL: [start.spring.io](https://start.spring.io)

이름:

위치:

프로젝트 이름(가) 다음에 생성됩니다. D:\project\2025\_springboot\thymeleafEx

☒ Git 저장소 생성

언어:

타입:

그룹:

아티팩트:

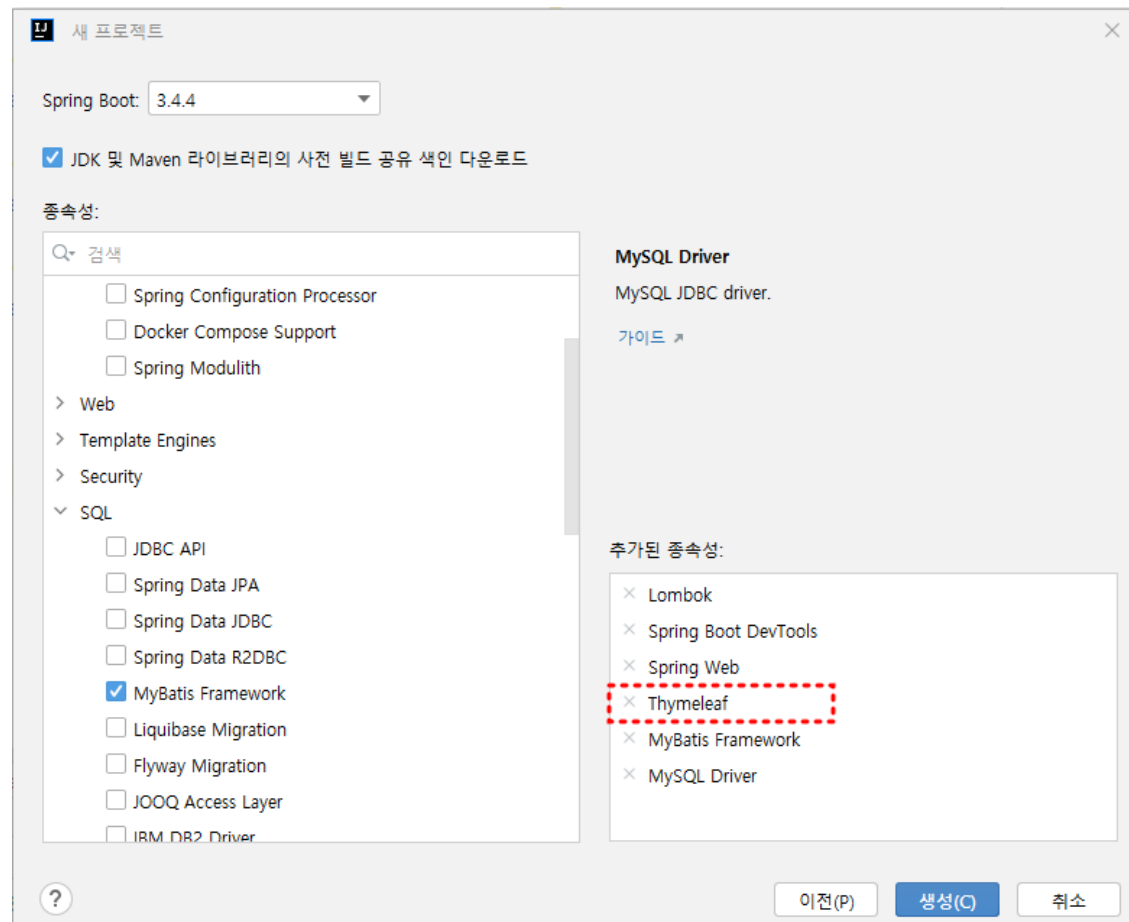
패키지 이름:

JDK:

Java:

패키지 생성:

[다음\(N\)](#) [취소](#)



새 프로젝트

Spring Boot:

☒ JDK 및 Maven 라이브러리의 사전 빌드 공유 색인 다운로드

종속성:

- ☐ Spring Configuration Processor
- ☐ Docker Compose Support
- ☐ Spring Modulith
- > Web
- > Template Engines
- > Security
- > SQL
  - ☐ JDBC API
  - ☐ Spring Data JPA
  - ☐ Spring Data JDBC
  - ☐ Spring Data R2DBC
  - ☒ MyBatis Framework
  - ☐ Liquibase Migration
  - ☐ Flyway Migration
  - ☐ JOOQ Access Layer
  - ☐ ORM Driver

MySQL Driver

MySQL JDBC driver.

[가이드](#)

추가된 종속성:

- × Lombok
- × Spring Boot DevTools
- × Spring Web
- × **Thymeleaf**
- × MyBatis Framework
- × MySQL Driver

[이전\(P\)](#) [생성\(C\)](#) [취소](#)

# 1. Thymeleaf Template Engines

## ❖ application.properties

```
#spring.application.name=thymeleafEx
server.port=8082

# database setting
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/springbootdb
spring.datasource.username=pgm
spring.datasource.password=1234

#log setting
logging.level.org.springframework=info
logging.level.com.example=debug

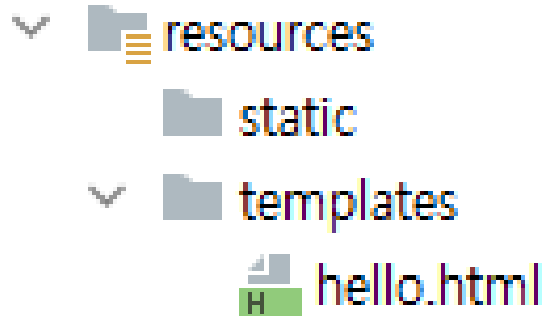
#xml mapper location
mybatis.mapper-locations=classpath:mappers/**/*.xml

dto location
mybatis.type-aliases-package=com.example.board01.dto
```

## 2. Thymeaf 기초

### ❖ Thymeleaf의 특징

- JSP의 경우 서블릿으로 변환된 후에 실행되는 방식
- Thymeleaf는 서버사이드 템플릿 엔진
- HTML의 구조에 추가적인 태그없이 선언적으로 데이터 바인딩 처리



```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>
  <h1 th:text="${msg}"> </h1>
  <h1>[[${msg}]]</h1>
</body>
</html>
```

## 2. Thymeleaf 기초

### ❖ Thymeleaf templates에 문자열 리턴

```
package com.example.thymeleafex.controller;

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

@Controller
public class SampleController {
    @GetMapping("/hello")
    public void hello(Model model){
        model.addAttribute("msg","Hello world");
    }
}
```

## 2. Thymeleaf 기초

### ❖ 주석처리

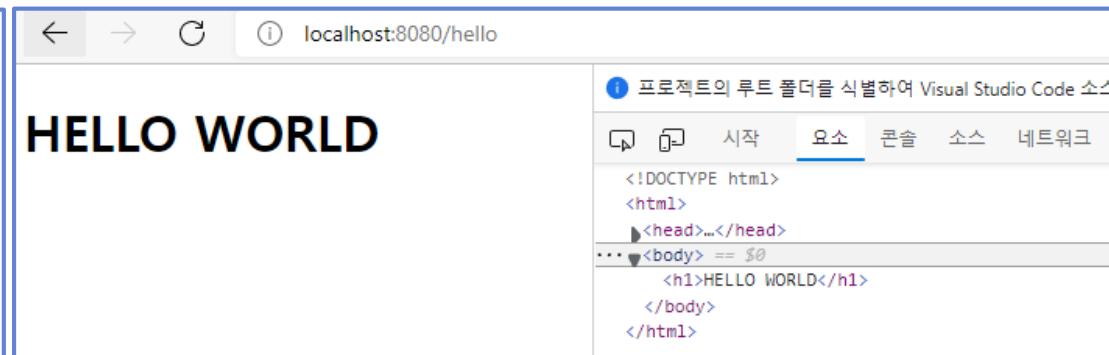
- 주석 처리를 해야 할 때에는 '`<!--/* ... */-->`' 를 이용

```
<body>
  <h1 th:text="{msg}"></h1>

  <!--/*  <h3 th:each="{sos}">SOS</h3> */-->

  <!--/*  ${aaaa + bbb } */-->
  <!--/*
    <div>
      <h1>AAAA</h1>
    </div>
  */-->

</body>
```



## 2. Thymeleaf 기초

### ❖ 배열

controller

SampleController

```
@GetMapping("/ex/ex1")
public void ex1(Model model){

    List<String> list = Arrays.asList("AAA","BBB","CCC","DDD");

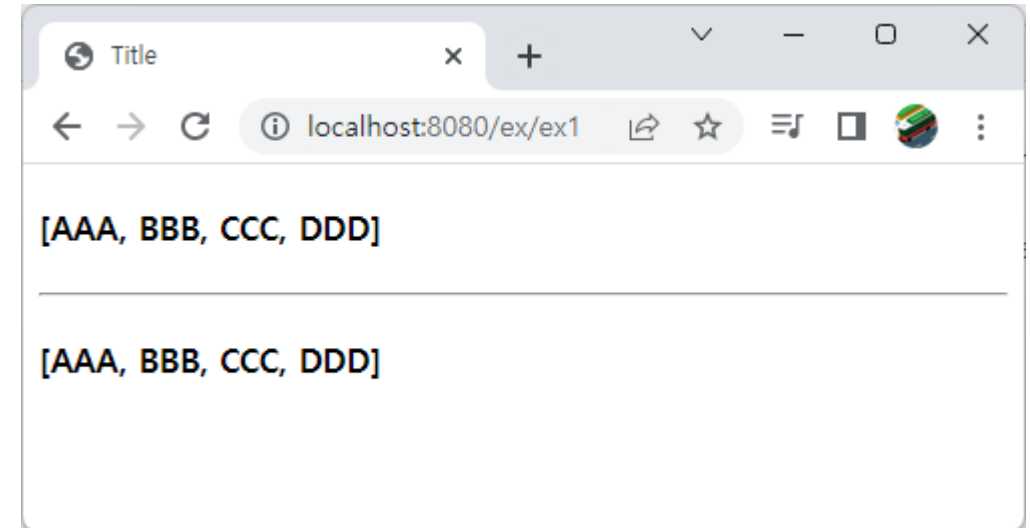
    model.addAttribute("list", list);
}
```

templates

ex

ex1.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>
  <h4>[[${list}]]</h4>
  <hr/>
  <h4 th:text="${list}"></h4>
</body>
</html>
```



### 3. 반복문과 제어문

#### ❖ th:with를 이용한 변수 선언

```
<div th:with="num1=${10}, num2=${20}">  
  <h4 th:text="${num1+num2}"></h4>  
</div>
```

#### ❖ th:if/th:unless 제어문 사용

```
<div th:with="num1=${10}, num2=${20}">  
  <span th:if="${num1>num2}">[[${num1}]]</span>  
  <span th:unless="${num1>num2}">[[${num1}]]</span>  
</div>
```



### 3. 반복문과 제어문

#### ❖ th:each 이용 배열/리스트/컬렉션 처리

```
<ul>
  <li th:each="str:${list}">
    <span>[[${str}]]</span> :
  </li>
</ul>
```

#### ❖ th:block 구문에 th:each 사용

```
<ul>
  <th:block th:each="str:${list}">
    <li>[[${str}]]</li>
  </th:block>
</ul>
```

# 3. 반복문과 제어문

## ❖ th:each 이용 배열/리스트/컬렉션 처리

- 반복문의 status 변수
  - 반복문에서 자주 사용하는 인덱스 번호나 개수등을 사용
  - index/count/size/first/odd/even

```
<ul>
  <li th:each="str,status: ${list}">
    <span th:if="${status.odd}"> ODD -- [[${str}]]</span>
    <span th:unless="${status.odd}"> EVEN -- [[${str}]]</span>
  </li>
</ul>
```

```
<ul>
  <li th:each="str, status:${list}">
    <span> [[${status.index}]]-[[${str}]]</span> :
    <span> [[${status.count}]]-[[${str}]]</span>
  </li>
</ul>
```

## 4. Thymeleaf를 이용한 링크 처리

### ❖ '@'를 이용한 경로 처리

- 절대 경로/컨텍스트 경로를 자동으로 처리
- 쿼리 스트링 처리

```
<a th:href="@{/hello1(name='AAA', age= 16)}">Go to /hello1</a><br/>
```

```
<a href="/hello1?name='AAA'&age=16">Go to/hell1</a>
```

```
<a th:href="@{/hello2(name='한글 처리', age= 16)}">Go to /hello2</a><br/>
```

```
<a href="/hello2?name='한글 처리'&age=16">Go to/hello2</a>
```

```
<a th:href="@{/hello3(types=${ {'AA','BB','CC'} }, age= 16)}">Go to /hello3</a><br/>
```

```
<a href="/hello3?types=AA&types=BB&types=CC">Go to/hello3</a>
```

## 4. Thymeleaf를 이용한 링크 처리

### ❖ Controller&View

```
@GetMapping("/hello1")
public void hello1(String name, int age, Model model){
    model.addAttribute("name",name);
    model.addAttribute("age",age);
}
```

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

```
@GetMapping("/hello3")
public void hello3(@RequestParam(name="types")
List<String> types, int age, Model model){
    model.addAttribute("types",types);
    model.addAttribute("age",age);
}
```

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

# 5. Thymeleaf의 JavaScript 인라인 처리

## ❖ 인라인 처리

- JavaScript의 경우 변수를 자동으로 JavaScript Object 형태로 출력

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>Title</title>
</head>
<body>
  <div th:text="${list}"> </div>
  <script th:inline="javascript">
    const list = [[${list}]]
    console.log(list)
  </script>
</body>
</html>
```

# 5. 5. Thymeleaf의 JavaScript 인라인 처리

## ❖ 인라인 처리: list

```
@GetMapping("ex/ex2")
public void ex2(Model model){
    log.info("ex2");
    List<String> strList2=new ArrayList<>();
    List<String> strList= IntStream.range(1,10)
        .mapToObj(i->"Data"+i)
        .collect(Collectors.toList());
    for(int i=1; i<10; i++){
        strList2.add("Data"+i);
    }
    model.addAttribute("strList", strList);
    model.addAttribute("strList2", strList2);
}
```

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org" lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<div>[[${strList}]]</div>
<div th:text="${strList}"></div>
<ul>
    <li th:each="str:${strList}">[[${str}]]</li>
</ul>
<ul>
    <th:block th:each="str:${strList}">
        <li>[[${str}]]</li>
    </th:block>
</ul>
<script th:inline="javascript">
    const list=[[${strList}]]
    console.log(list)
</script>
</body>
</html>
```

# 5. Thymeleaf의 JavaScript 인라인 처리

## ❖ 인라인 처리: map, object

Controller

```
@GetMapping("ex/ex2")
public void ex2(Model model){
    ...생략

    Map<String, Integer> maps=new HashMap<>();
    maps.put("홍길동",80);
    maps.put("박경미",75);
    maps.put("윤요섭",85);
    model.addAttribute("maps",maps);

    SampleDTO sampleDTO=new SampleDTO();
    sampleDTO.setName("hong");
    sampleDTO.setAge(20);
    sampleDTO.setGender("남자");

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

DTO 클래스

```
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class SampleDTO {
    private String name;
    private int age;
    private String gender;
}
```

# 5. Thymeleaf의 JavaScript 인라인 처리

## ❖ 인라인 처리 : template, 브라우저 console에서 확인 결과

html 파일

.. 생략

```
<div th:text="${maps}"></div>
<div>[${maps}]</div>
<div th:text="${maps.홍길동}"></div>
<div>[${sampleDTO}]</div>
<div th:text="${sampleDTO}"></div>
<div>[${sampleDTO.name}]</div>
```

```
<script th:inline="javascript">
  const list=[[${strList}]]
  const map=[[${maps}]]
  const dto=[[${sampleDTO}]]
  console.log(list)
  console.log(map)
  console.log(dto)
</script>
```

localhost:8082/ex/ex02

[Data1, Data2, Data3, Data4, Data5, Data6, Data7, Data8, Data9]  
[Data1, Data2, Data3, Data4, Data5, Data6, Data7, Data8, Data9]

- Data1
- Data2
- Data3
- Data4
- Data5
- Data6
- Data7
- Data8
- Data9

{홍길동=80, 박경미=75, 윤요섭=85}  
{홍길동=80, 박경미=75, 윤요섭=85}  
80  
SampleDTO(name=hong, age=20, gender=남자)  
SampleDTO(name=hong, age=20, gender=남자)  
hong

DevTools is now available in Korean! Always match Chrome's language Switch DevTools to Korean

Elements Console Sources Network Performance Memory Application Pri

top Filter

(9) ['Data1', 'Data2', 'Data3', 'Data4', 'Data5', 'Data6', 'Data7', 'Data8', 'Data9']

- 0: "Data1"
- 1: "Data2"
- 2: "Data3"
- 3: "Data4"
- 4: "Data5"
- 5: "Data6"
- 6: "Data7"
- 7: "Data8"
- 8: "Data9"
- length: 9

[[Prototype]]: Array(0)

{홍길동: 80, 박경미: 75, 윤요섭: 85}

- 박경미: 75
- 윤요섭: 85
- 홍길동: 80

[[Prototype]]: Object

{name: 'hong', age: 20, gender: '남자'}

- age: 20
- gender: "남자"
- name: "hong"

[[Prototype]]: Object



## 6. 레이아웃 기능

---

### ❖ 레이아웃 기능(layout feature)

- 여러 페이지에서 공통되는 레이아웃(헤더, 푸터, 네비게이션 등)을 재사용할 수 있게 해줌

### ❖ 종속성 추가

```
implementation 'nz.net.ultraq.thymeleaf:thymeleaf-layout-dialect:3.2.0'
```

# 6. 레이아웃 기능

## ❖ 레이아웃 파일 작성

▼ templates

> ex

▼ layout

layout1.html

```
<!DOCTYPE html>
<html xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      xmlns:th="http://www.thymeleaf.org">
<head>
  <title>Layout page</title>
</head>
<body>

<div>
  <h3>Sample Layout Header</h3>
</div>

<div layout:fragment="content">
  <p>Page content goes here</p>
</div>

<div>
  <h3>Sample Layout Footer</h3>
</div>

<th:block layout:fragment="script" >
</th:block>

</body>
</html>
```

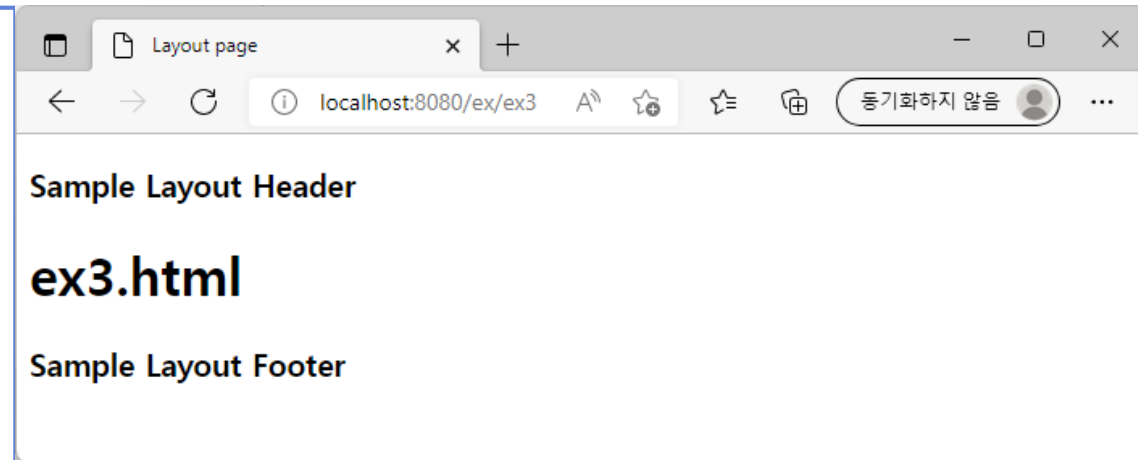
# 6. 레이아웃 기능

## ❖ 레이아웃 적용

- <th:block>을 이용해서 필요한 부분만을 작성하는 방식
  - layout:fragment를 이용해서 변경이 가능한 부분을 지정하고 나중에 다른 내용물로 변경 가능

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      layout:decorate="~{layout/layout1.html}" >

<div layout:fragment="content">
  <h1>ex3.html</h1>
</div>
```



## 7. thymeleaf template 적용 실습

❖ board01 프로젝트의 jsp view를 thymeleaf로 변경하여 적용하기(p476 ~참고)

❖ 템플릿 다운로드

- url : <https://startbootstrap.com/template/simple-sidebar>
- Free Download 클릭하여 다운로드 후 압축해제 후 assets, css, js, index.html을 static에 복사

