

## 9장. Thymeleaf 페이지 레이아웃



*Thymeleaf*



Spring Boot

- **Thymeleaf**

Thymeleaf는 웹뿐만 아니라 다른 환경을 위한 최신의 서버-사이드 자바 Template Engine(템플릿 엔진) 또는 view 엔진이며, HTML, CSS, XML, JS 및 TEXT까지 수용(이는 다른 템플릿 엔진과 동일)한다.

타임리프의 주 목표는 유지관리가 쉬운 템플릿 생성 방법을 제공하는 것이며, 실제로 템플릿에 영향을 주지 않는(**HTML의 구조를 깨지 않는, 기존 HTML 코드를 변경하지 않고 덧붙이는 코드**) 방식을 사용한다.



# Thymeleaf(타임 리프)

- 컨트롤러 작성

```
@Slf4j
@RequestMapping("/sample")
@Controller
public class SampleController {

    @GetMapping("/ex1")
    public void ex1() {
        log.info("ex1.....");
    }

    //상품 1개 정보
    @GetMapping("/ex2")
    public String ex2(Model model) {
        ItemDto itemDto = new ItemDto();
        itemDto.setItemNm("상품1");
        itemDto.setPrice(20000);
        itemDto.setItemDetail("상품1 상세 설명");
        itemDto.setRegTime(LocalDateTime.now());

        model.addAttribute("itemDto", itemDto);
        return "sample/ex2";
    }
}
```



# Thymeleaf(타임 리프)

- 컨트롤러 작성

```
//상품 목록
@GetMapping("/ex3")
public String ex3(Model model) {
    List<ItemDto> itemDtoList = new ArrayList<>();

    for(int i=1; i<=10; i++) {
        ItemDto itemDto = new ItemDto();
        itemDto.setId((long) i);
        itemDto.setItemNm("상품" + i);
        itemDto.setPrice(1000*i);
        itemDto.setItemDetail("상품" + i + "상세 설명");
        itemDto.setRegTime(LocalDateTime.now());

        itemDtoList.add(itemDto);
    }
    model.addAttribute("itemDtoList", itemDtoList);
    return "sample/ex3";
}
```



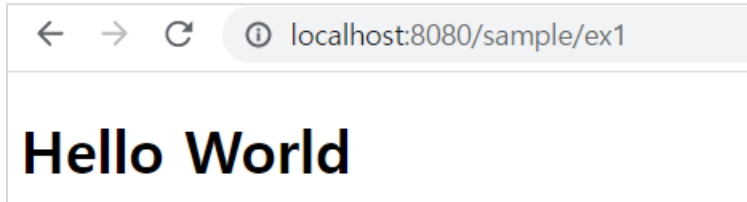
- 컨트롤러 작성

```
@GetMapping("/ex4")  
public String ex4() {  
    return "sample/ex4";  
}  
  
@GetMapping("/ex5")  
public String ex5() {  
    return "sample/ex5";  
}
```



# Thymeleaf(타임 리프)

- 화면(view)



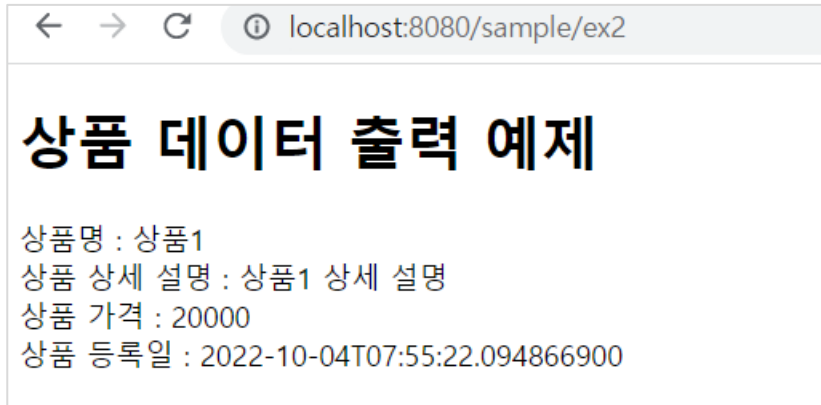
**th:text = "\${data}" : 데이터 출력**

```
<title>Title</title>
</head>
<body>
  <h1 th:text="${'Hello World'}"></h1>
</body>
</html>
```

ex1.html



- 화면(view)



ex2.html

```
<h1>상품 데이터 출력 예제</h1>
<div>
  상품명 : <span th:text="${itemDto.itemNm}"></span>
</div>
<div>
  상품 상세 설명 : <span th:text="${itemDto.itemDetail}"></span>
</div>
<div>
  상품 가격 : <span th:text="${itemDto.price}"></span>
</div>
<div>
  상품 등록일 : <span th:text="${itemDto.regTime}"></span>
</div>
```



- 화면(view)

← → ↺ ⓘ localhost:8080/sample/ex3

## 상품 리스트 출력 예제

순번	상품명	가격	상품설명	등록일	등록일
2	상품2	2000	상품2상세 설명	2022-10-04 07:56:25	2022-10-04 07:56:25
4	상품4	4000	상품4상세 설명	2022-10-04 07:56:25	2022-10-04 07:56:25
6	상품6	6000	상품6상세 설명	2022-10-04 07:56:25	2022-10-04 07:56:25
8	상품8	8000	상품8상세 설명	2022-10-04 07:56:25	2022-10-04 07:56:25
10	상품10	10000	상품10상세 설명	2022-10-04 07:56:25	2022-10-04 07:56:25





# Thymeleaf(타임 리프)

**th:each** = “변수: \${목록}” – 데이터를 반복 처리

**[[ \${itemDto.id} ]]**은 인라인 표현식

**th:if** = “조건절” – 조건문 처리



# Thymeleaf(타임 리프)

<h1>상품 리스트 출력 예제</h1>

ex3.html

<table border="1">

<thead>

<tr>

<th>순번</th><th>상품명</th><th>가격</th><th>상품설명</th><th>등록일</th><th>등록일</th>

</tr>

</thead>

<tbody>

<tr th:each="itemDto : \${itemDtoList}" th:if="\${itemDto.id} % 2 == 0">

<td th:text="\${itemDto.id}">

<td th:text="\${itemDto.itemNm}">

<td th:text="\${itemDto.price}">

<td th:text="\${itemDto.itemDetail}">

<!--<td th:text="\${itemDto.regTime}">-->

<td th:text="\${#temporals.format(itemDto.regTime, 'yyyy-MM-dd HH:mm:ss')}">

<!--인라인 표기-->

<td>[[\${#temporals.format(itemDto.regTime, 'yyyy-MM-dd HH:mm:ss')}]</td>

</tr>

</tbody>

</table>



# Thymeleaf(타임 리프)

- 화면(view)



ex4.html

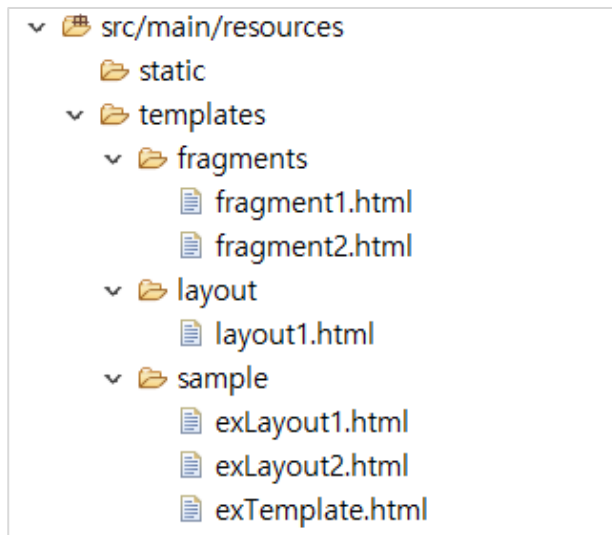
```
<h1>Thymeleaf 링크 처리 예제</h1>
<div>
  <a th:href="@{/sample/ex1}">예제1 페이지 이동</a>
</div>
<div>
  <a th:href="@{http://www.thymeleaf.org}">타임리프 공식 페이지 이동</a>
</div>
```



## ▪ Thymeleaf 페이지 레이아웃

### 1) include 방식의 처리

특정 부분을 다른 내용으로 변경할 수 있는 태그는 th:replace 또는 th:insert가 사용된다. th:replace는 기존의 내용을 완전히 대체하는 방식이고, th:insert는 기존의 바깥쪽 태그는 유지하면서 추가되는 방식이라는 차이가 있다.



templates 아래에 fragments 폴더 생성후  
fragment1.html 파일을 만듦.

**이 파일을 사용하기 위해**

sample 폴더 아래에 exLayout1.html을 생성



- SampleController의 일부

```
@Slf4j
@RequestMapping("/sample/")
@Controller
public class SampleController {

    @GetMapping("/exLayout1")
    public void exLayout1() {
        log.info("exLayout1.....");
    }
}
```



- fragment1.html

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

    <div th:fragment="part1">
        <h2>Part1 </h2>
    </div>

    <div th:fragment="part2">
        <h2>Part2 </h2>
    </div>

    <div th:fragment="part3">
        <h2>Part3 </h2>
    </div>

</html>
```



- exLayout1.html

fragment1.html의 조각들을 가져와서 사용하는 코드 작성

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

    <h2>Layout 1 - 1 </h2>
    <div th:replace="~/fragments/fragment1 :: part1">
    </div>

    <h2>Layout 1 - 2 </h2>
    <div th:replace="~/fragments/fragment1 :: part2">
    </div>

    <h2>Layout 1 - 3 </h2>
    <div th:insert="~/fragments/fragment1 :: part3">
    </div>

</html>
```



- 출력 화면





- fragment2.html

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

    <div>
        <h2>Fragments2 File</h2>
        <h2>Fragments2 File</h2>
        <h2>Fragments2 File</h2>
    </div>

</html>
```



# Thymeleaf(타임 리프)

- exLayout1.html -> fragment2.html 전체 가져오기

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

    <!-- fragment2 파일 전체 포함하기 -->
    <div style="border: 1px solid #00f">
        <th:block th:replace="~/fragments/fragment2">
        </th:block>
    </div>

    <h2>Layout 1 - 1 </h2>
    <div th:replace="~/fragments/fragment1 :: part1">
    </div>

    <h2>Layout 1 - 2 </h2>
    <div th:replace="~/fragments/fragment1 :: part2">
    </div>
```



- 출력 화면

Fragments2 File

Fragments2 File

Fragments2 File

Layout 1 - 1

Part1

Layout 1 - 2

Part2

Layout 1 - 3

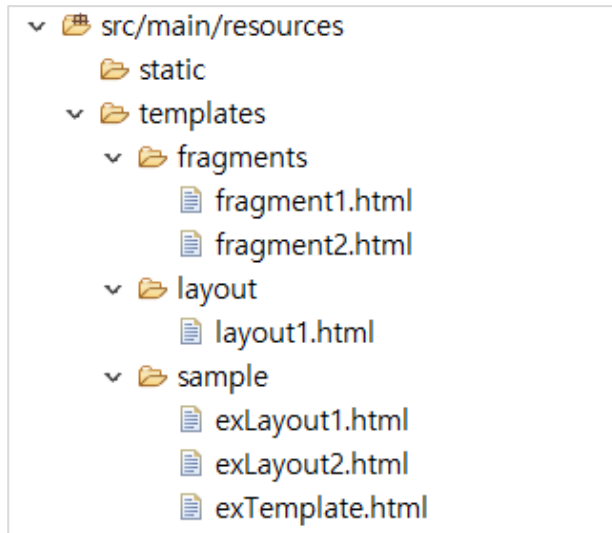
Part3



## ▪ Thymeleaf 페이지 레이아웃

### 2) 파라미터 방식의 처리

특정한 태그를 파라미터처럼 전달해서 다른 th:fragment에서 사용할 수 있다.



templates 아래에 layout폴더 생성후  
layout1.html 파일을 만듦.

이 파일을 사용하기 위해

sample 폴더 아래에 exTemplate.html을 생성



# Thymeleaf(타임 리프)

- SampleController의 일부..

```
@Slf4j
@RequestMapping("/sample/")
@Controller
public class SampleController {

    /*@GetMapping("/exLayout1")
    public void exLayout1() {
        log.info("exLayout1.....");
    }*/

    //타임리프 Layout
    @GetMapping({"{/exLayout1", "/exTemplate"})
    public void exLayout() {
        Log.info("exLayout.....");
    }
}
```



# Thymeleaf(타임 리프)

## ▪ layout1.html

```
<html xmlns:th="http://www.thymeleaf.org">
  <style>
    *{margin: 0; padding: 0;}
    .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>

  <div class="header">
    <h2>HEADER</h2>
  </div>

  <!-- 본문 영역 -->
  <div class="content">
    <h2>CONTENT</h2>
  </div>

  <div class="footer">
    <h2>FOOTER</h2>
  </div>
</html>
```



- layout1.html



# Thymeleaf(타임 리프)

- layout1.html 추가

중간에 "CONTENT" 로 표시된 영역이 다른 내용으로 변경되어야 함

```
<html xmlns:th="http://www.thymeleaf.org">
<!-- 본문 영역의 ${content}를 파라미터로 받도록 설정 -->
<th:block th:fragment="setContent(content)">
    <style>
        *{margin: 0; padding: 0;}
    </style>
    <!-- 본문 영역 -->
    <div class="content">
        <th:block th:replace="${content}">
            </th:block>
        </div>
    </th:block>
    <div class="footer">
        <h2>FOOTER</h2>
    </div>
</html>
```





- exTemplate.html

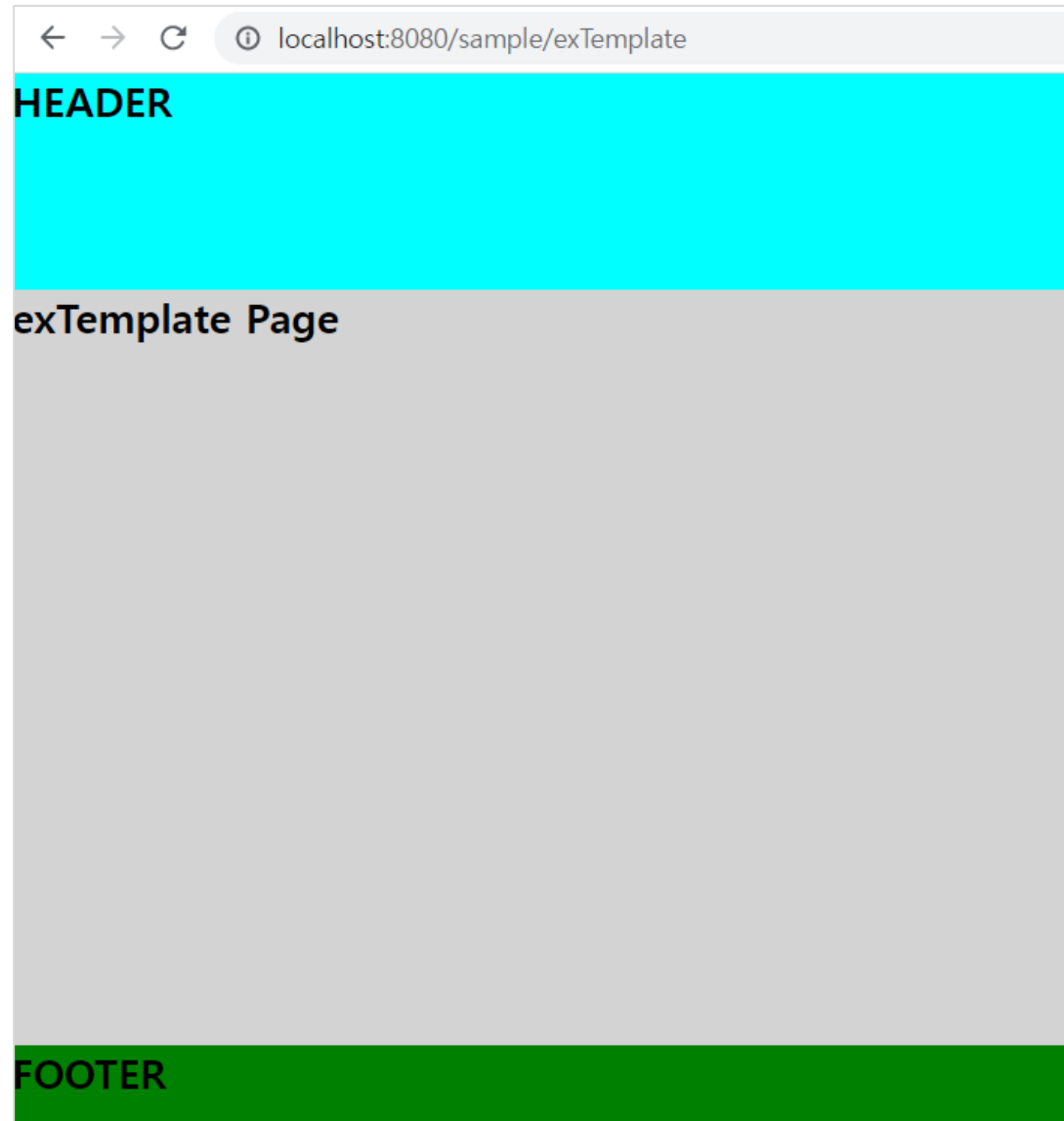
```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<th:block th:replace="~/layout/layout1 :: setContent(~{this::content})">

    <th:block th:fragment="content">
        <h2>exTemplate Page</h2>
    </th:block>

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



- 출력 화면



# Thymeleaf(타임 리프)

- layout1.css 파일로 스타일 적용하기

```
@charset "UTF-8";

*{margin: 0; padding: 0;}
.header{width:100vw; height: 20vh; background-color: aqua;}
.content{width:100vw; height: 70vh; background-color: lightgray;}
.footer{width:100vw; height: 10vh; background-color: yellowgreen;}
```

```
<html xmlns:th="http://www.thymeleaf.org">
<!-- 본문 영역의 ${content}를 파라미터로 받도록 설정 -->
<th:block th:fragment="setContent(content)">
    <!-- layout.css 파일 포함 -->
    <link th:href="@{/css/layout1.css}" rel="stylesheet">

    <div class="header">
        <h2>HEADER</h2>
    </div>
```



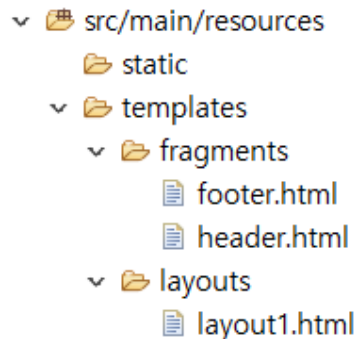
# Thymeleaf(타임 리프)

## ▪ Thymeleaf 페이지 레이아웃

**Thymeleaf Layout Dialect dependency** 추가하기

- 하나의 레이아웃을 여러 페이지에 똑같이 적용할 수 있다.

```
<dependency>
  <groupId>nz.net.ultraq.thymeleaf</groupId>
  <artifactId>thymeleaf-layout-dialect</artifactId>
</dependency>
```



src/main/resources

- static
- templates
  - fragments
    - footer.html
    - header.html
  - layouts
    - layout1.html

templates 아래에 fragments 폴더 생성후 header.html과 footer.html을 생성.

다음 layouts 폴더를 만들고 layout1.html 파일을 생성함



# Thymeleaf(타임 리프)

- Thymeleaf 페이지 레이아웃

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

    <div th:fragment="header">
        header 영역 입니다.
    </div>

</html>
```

header.html

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

    <div th:fragment="footer">
        footer 영역 입니다.
    </div>

</html>
```

footer.html



# Thymeleaf(타임 리프)

- layout1 파일을 기본(표준) 템플릿이라고도 함

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout">
<head>
<meta charset="UTF-8">
<title>Title</title>
    <th:block layout:fragment="script"></th:block>
    <th:block layout:fragment="css"></th:block>
</head>
<body>
    <div th:replace="fragments/header::header"></div>

    <div layout:fragment="content">

    </div>

    <div th:replace="fragments/footer::footer"></div>
</body>
</html>
```

layout1.html



# Thymeleaf(타임 리프)

- Thymeleaf 페이지 레이아웃

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

    <div layout:fragment="content">
        본문 영역입니다.
    </div>
</html>
```

ex5.html

